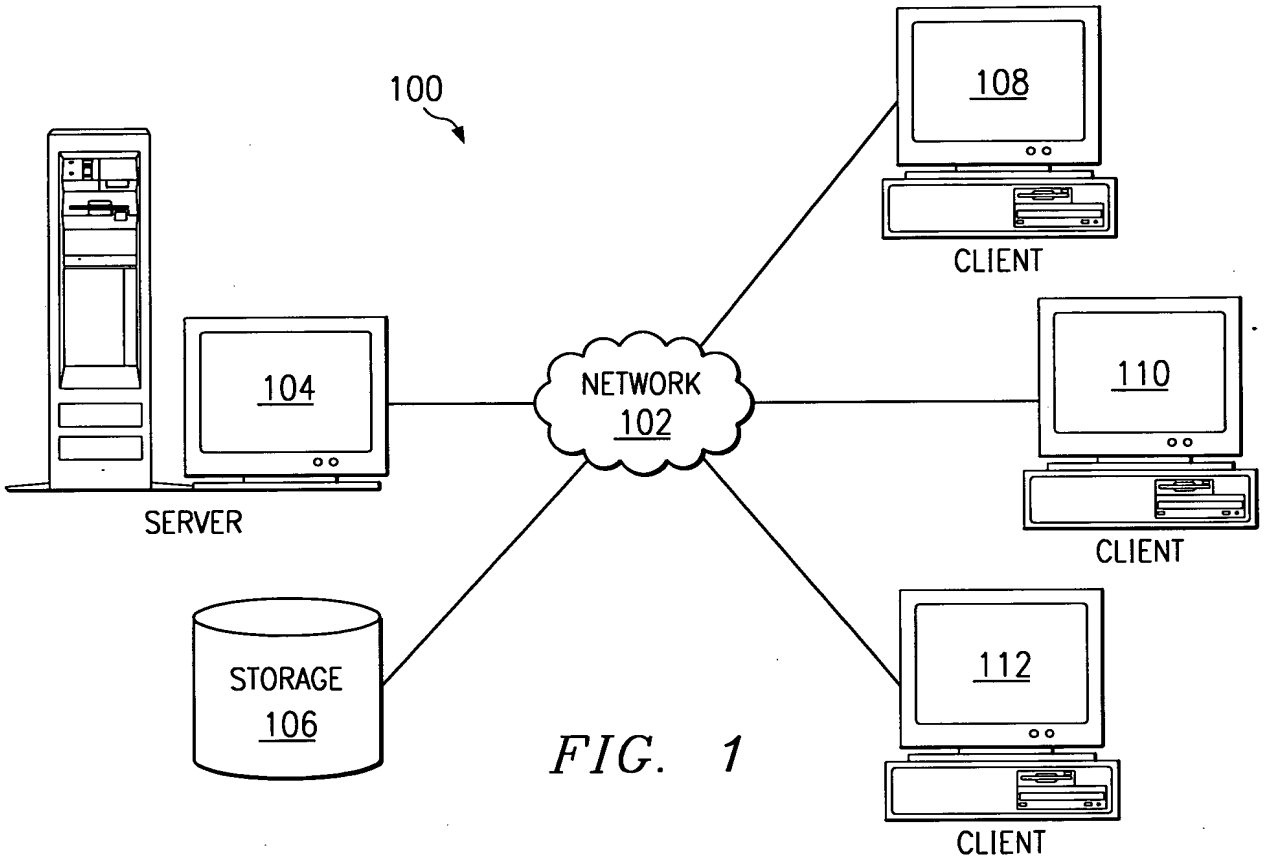

100
108
CLIENT
104
SERVER
NETWORK
102
110
CLIENT
STORAGE
106
112
CLIENT

FIG. 1

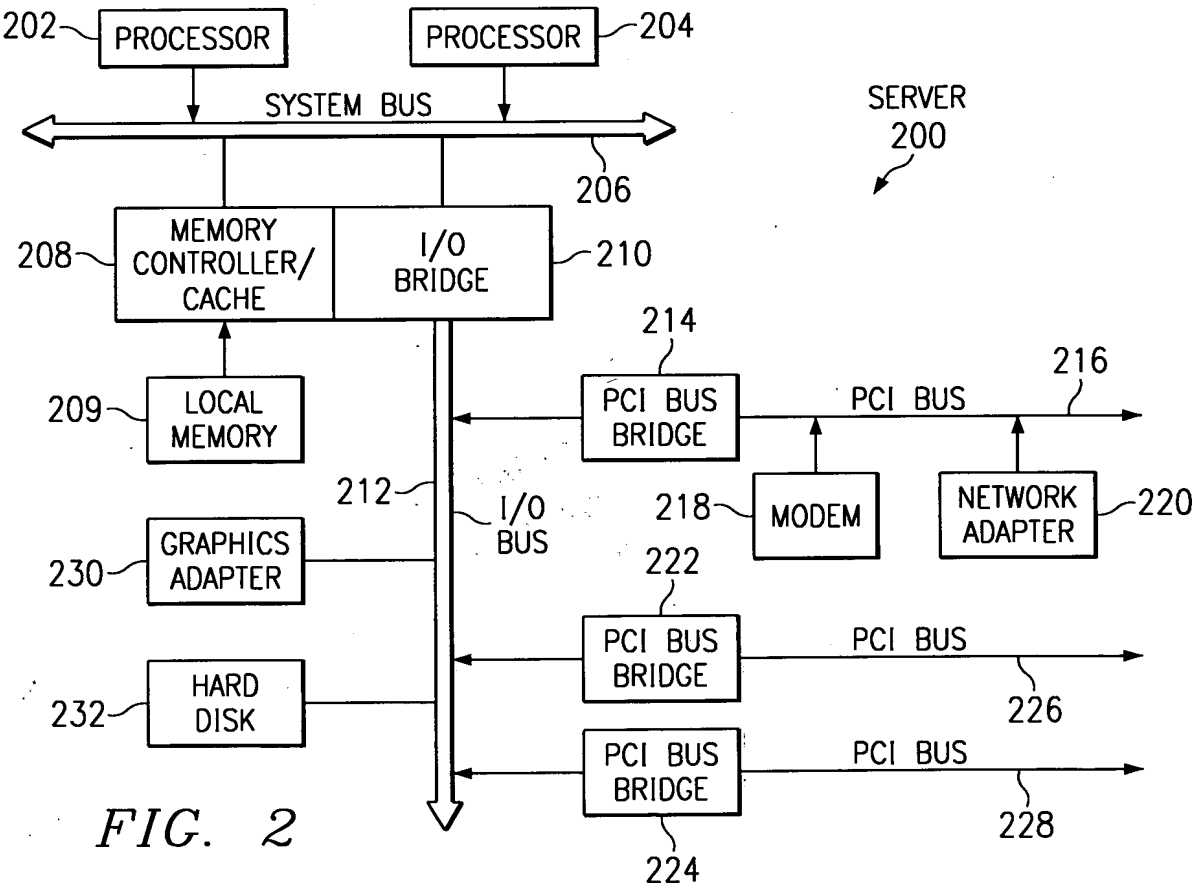

202
PROCESSOR
PROCESSOR
204
SYSTEM BUS
206
SERVER
200
208
MEMORY CONTROLLER/ CACHE
I/O BRIDGE
210
214
216
209
LOCAL MEMORY
PCI BUS BRIDGE
PCI BUS
212
I/O BUS
218
MODEM
NETWORK ADAPTER
220
230
GRAPHICS ADAPTER
222
PCI BUS BRIDGE
PCI BUS
226
232
HARD DISK
PCI BUS BRIDGE
PCI BUS
228
224

FIG. 2

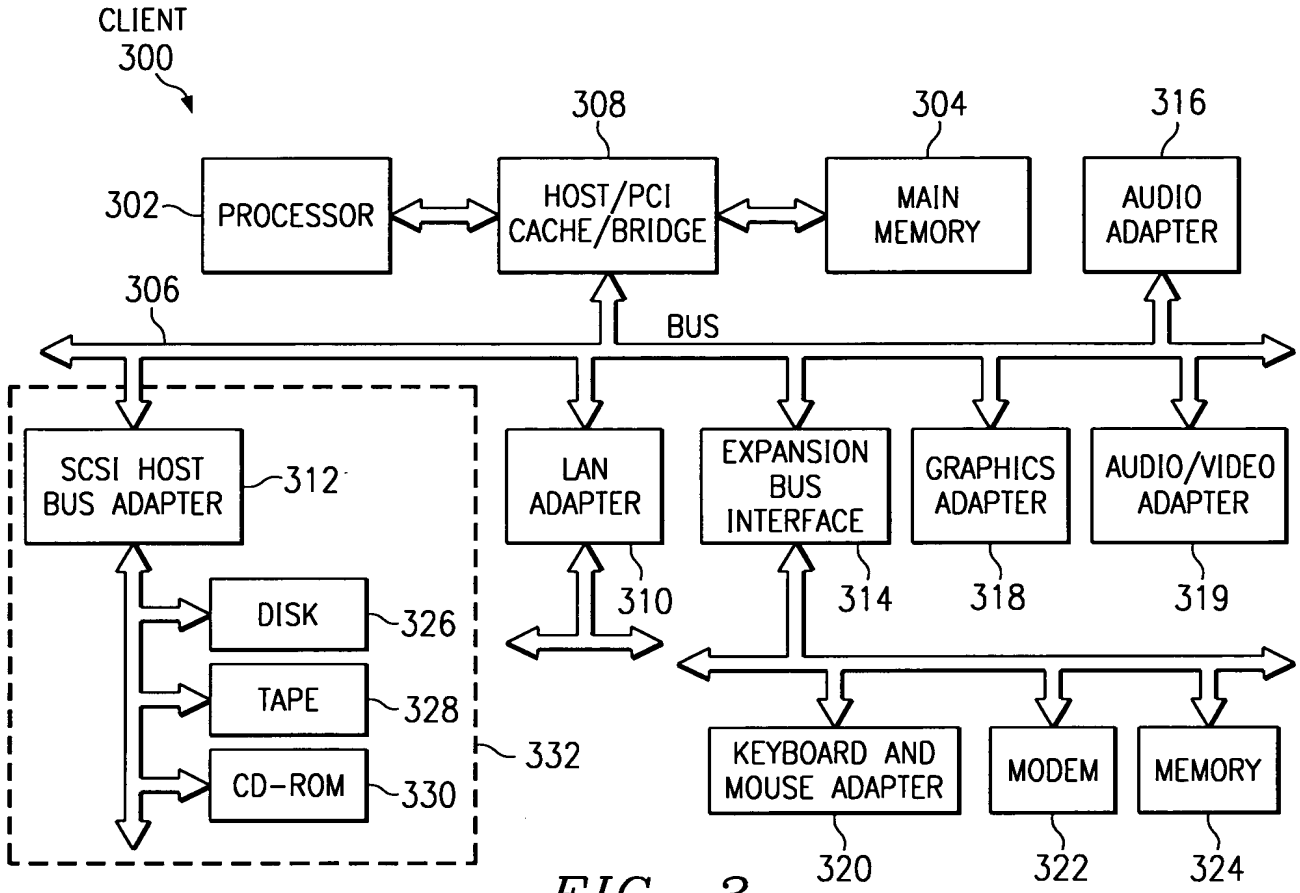

CLIENT
300
308
304
316
302
PROCESSOR
HOST/PCI
CACHE/BRIDGE
MAIN
MEMORY
AUDIO
ADAPTER
306
BUS
SCSI HOST
BUS ADAPTER
312
LAN
ADAPTER
EXPANSION
BUS
INTERFACE
GRAPHICS
ADAPTER
AUDIO/VIDEO
ADAPTER
310
314
318
319
DISK
326
TAPE
328
CD-ROM
330
332
KEYBOARD AND
MOUSE ADAPTER
MODEM
MEMORY
320
322
324

*FIG. 3*

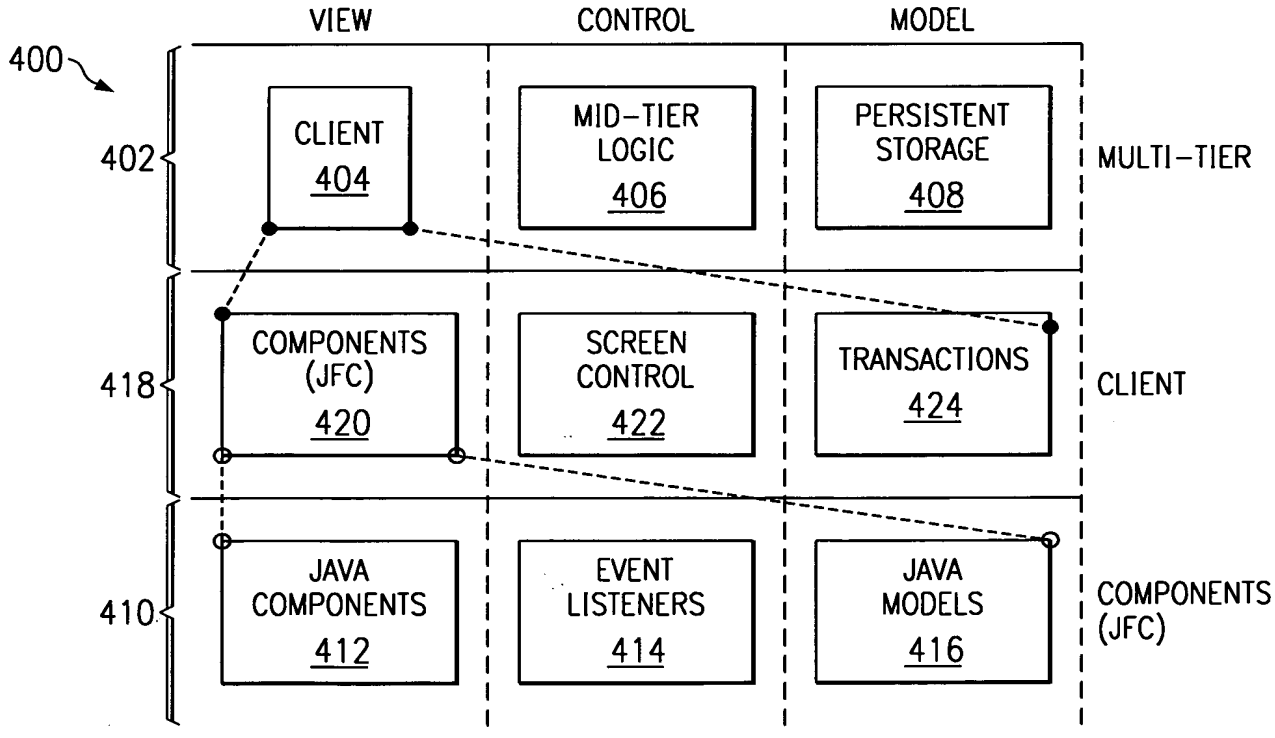
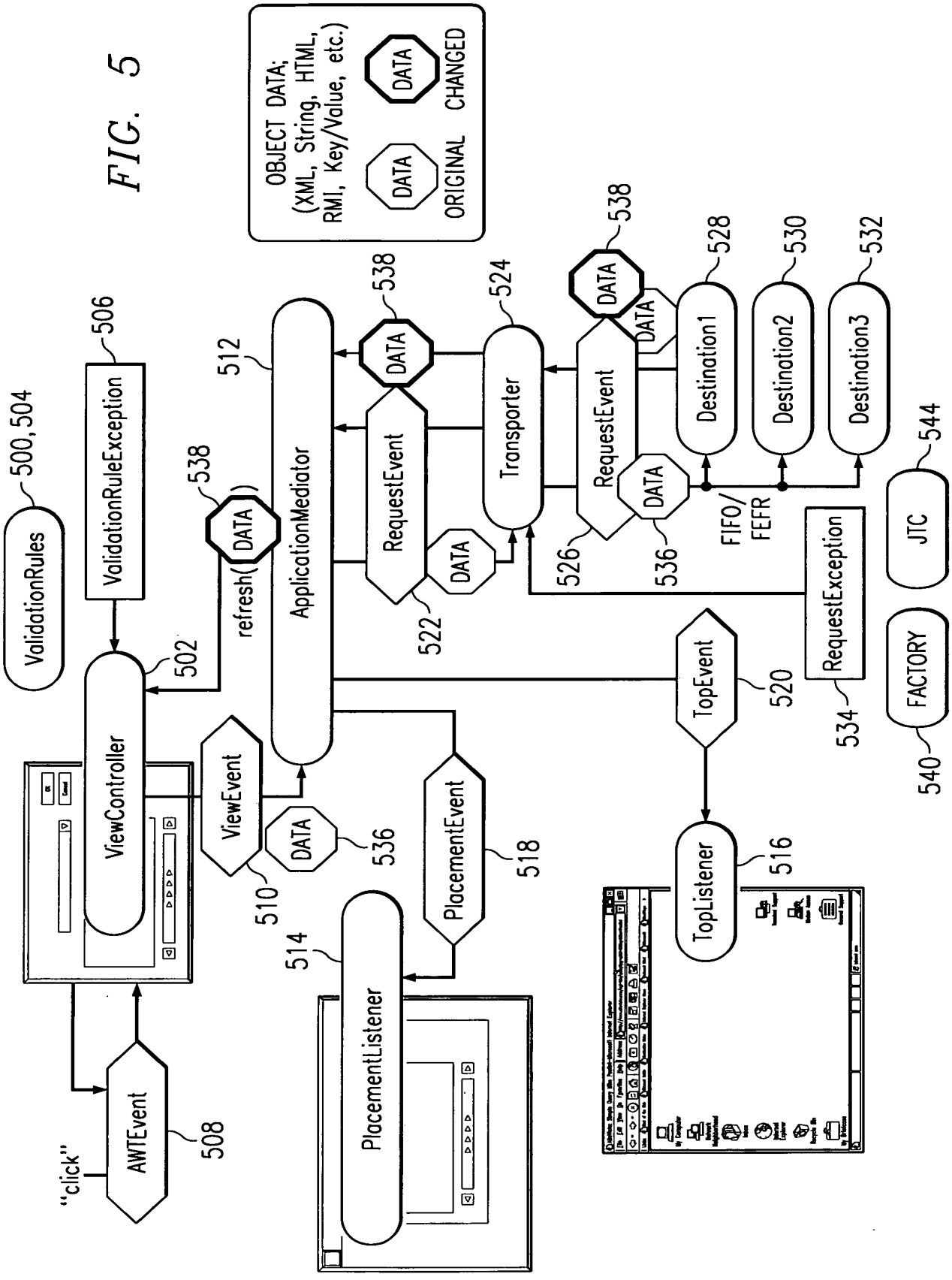

VIEW
CONTROL
MODEL
400
402
CLIENT
404
MID-TIER
LOGIC
406
PERSISTENT
STORAGE
408
MULTI-TIER
418
COMPONENTS
(JFC)
420
SCREEN
CONTROL
422
TRANSACTIONS
424
CLIENT
410
JAVA
COMPONENTS
412
EVENT
LISTENERS
414
JAVA
MODELS
416
COMPONENTS
(JFC)

*FIG. 4*

FIG. 5


OBJECT DATA;
(XML, String, HTML,
RMI, Key/Value, etc.)
DATA
ORIGINAL
DATA
CHANGED
500, 504
ValidationRules
506
ValidationRuleException
538
DATA
refresh( )
512
ApplicationMediator
538
DATA
RequestEvent
522
DATA
524
Transporter
538
DATA
RequestEvent
526
DATA
536
DATA
528
Destination1
530
Destination2
532
Destination3
FIFO/
FEFR
534
RequestException
544
JTC
540
FACTORY
520
TopEvent
516
TopListener
502
ViewController
510
ViewEvent
DATA
536
514
PlacementListener
518
PlacementEvent
"click"
AWTEvent
508

600

Class Hierarchy

```
class java.lang.Object
    interface com.ibm.jtc.ApplicationMediator (extends com.ibm.jtc.JTC)
    class com.ibm.jtc.ApplicationMediatorImpl (implements com.ibm.jtc.ApplicationMediator,
    com.ibm.jtc.ViewListener, com.ibm.jtc.RequestResponseListener)
    interface com.ibm.jtc.Destination (extends com.ibm.jtc.JTC)
    class com.ibm.jtc.DestinationImpl (implements com.ibm.jtc.Destination)
    class java.util.EventObject (implements java.io.Serializable)
      class com.ibm.jtc.PlacementEvent (implements java.io.Serializable)
      class com.ibm.jtc.RequestEvent (implements java.io.Serializable)
      class com.ibm.jtc.TopEvent (implements java.io.Serializable)
      class com.ibm.jtc.ViewEvent (implements java.io.Serializable)
    class com.ibm.jtc.Factory (implements java.io.Serializable)
    interface com.ibm.jtc.JTC (extends java.io.Serializable)
    interface com.ibm.jtc.PlacementListener
    interface com.ibm.jtc.RequestListener
    interface com.ibm.jtc.RequestResponseListener
    class java.lang.Throwable (implements java.io.Serializable)
      class java.lang.Exception
        class com.ibm.jtc.RequestException (implements java.io.Serializable)
        class com.ibm.jtc.ValidationRuleException (implements java.io.Serializable)
    interface com.ibm.jtc.TopListener
    class com.ibm.jtc.Transporter (implements com.ibm.jtc.RequestListener, com.ibm.jtc.JTC)
    class com.ibm.jtc.ValidationRule (implements java.io.Serializable)
    interface com.ibm.jtc.ViewController (extends com.ibm.jtc.JTC)
    interface com.ibm.jtc.ViewListener
```

*FIG. 6*

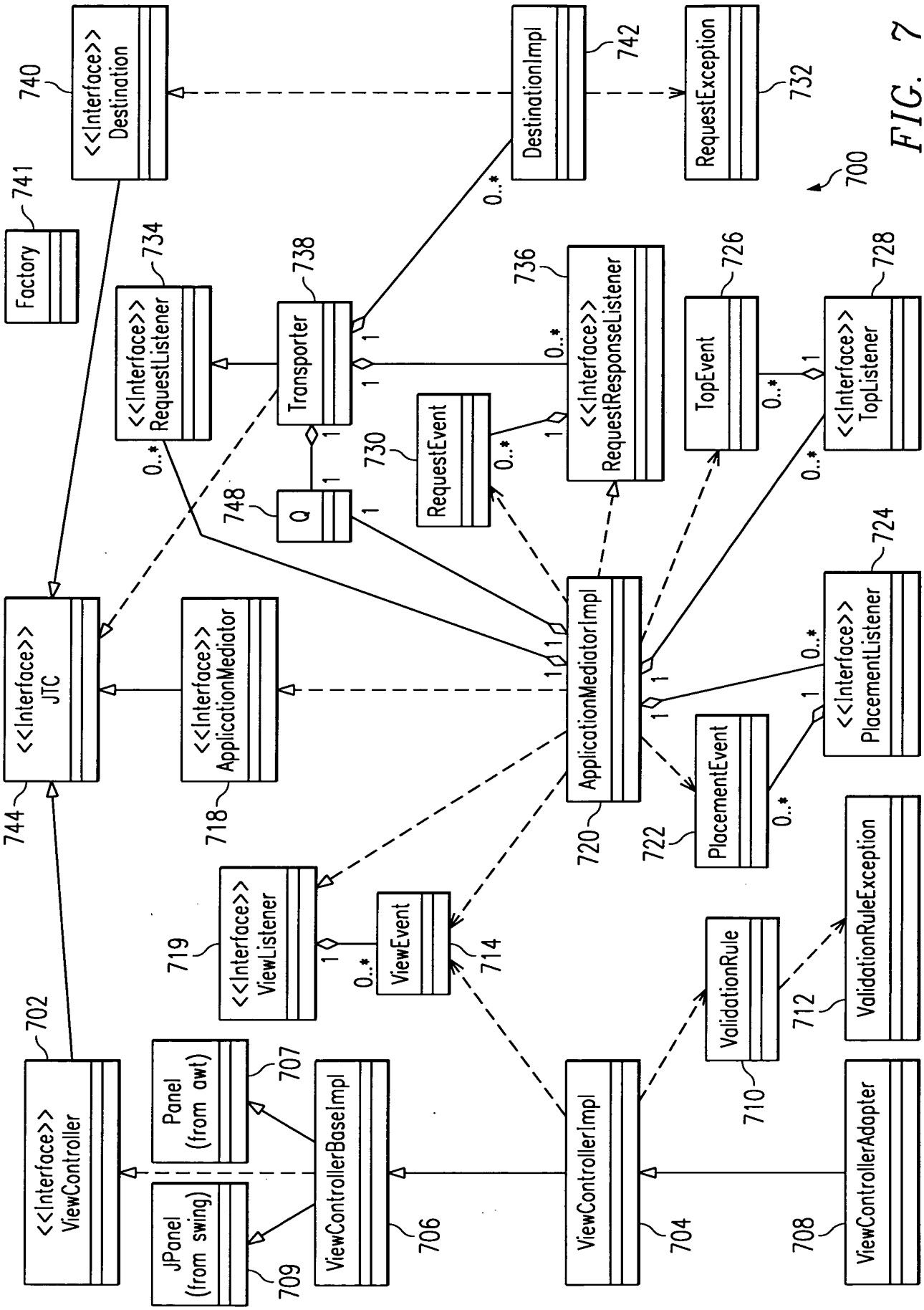

<<Interface>>
Destination
740
DestinationImpl
742
RequestException
732
700
Factory
741
<<Interface>>
RequestListener
734
Transporter
738
<<Interface>>
RequestResponseListener
736
TopEvent
726
<<Interface>>
TopListener
728
Q
748
RequestEvent
730
<<Interface>>
JTC
744
<<Interface>>
ApplicationMediator
718
ApplicationMediatorImpl
720
<<Interface>>
PlacementListener
724
PlacementEvent
722
<<Interface>>
ViewListener
719
ViewEvent
714
ValidationRule
710
ValidationRuleException
712
<<Interface>>
ViewController
702
Panel
(from awt)
707
JPanel
(from swing)
709
ViewControllerBaseImpl
706
ViewControllerImpl
704
ViewControllerAdapter
708
0..*
1

FIG. 7

ViewController

FIG. 8A

800

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String _copyright | (c) International Business Machines, Inc., 1997 1998 1999. All rights reserved. |

FIG. 8B

802

Methods

| Name | Declaration | Description |
|---|---|---|
| addViewListener | public abstract void addViewListener (ViewListener listener) | Invoked when a ViewListener is added. |
| getComponent | public abstract Component getComponent() | Invoked when the ViewController as a component is needed. |
| getPermissions | public abstract String[] getPermissions () | Invoked when the ViewController permission keys are needed. |
| isValid | public abstract boolean isValid() | Invoked when a ViewController's GUI state needs to be checked to see if it is valid. |
| isVisible | public abstract boolean isVisible() | Invoked to see if the ViewController is visible. |
| refresh | public abstract void refresh (Object data) | Invoked to supply new or changed data. |
| removeViewListener | public abstract void removeViewListener (ViewListener listener) | Invoked to remove a ViewListener. |
| setPermissions | public abstract void setPermissions (Hashtable permissions) | Invoked to set the permissions keys and values. |
| setProperties | public abstract void setProperties (Properties properties) | Invoked to set the properties. |
| setResources | public abstract void setResources (ResourceBundle bundle) | Invoked to set the resources. |
| setValidationLevel | public abstract void setValidationLevel (int level) | Invoked to give a hint to the ViewController as to what validation level to use. The value for level defined in this interface include: NONE=try to do no validation EVENT=try to do validation every event (key) FOCUS=try to do validation on focus change VIEWEVENT=try to do validation before a ViewEvent is issued. |
| setVisible | public abstract void setVisible (boolean visible) | Invoked to set the visibility. |

ViewControllerImpl

900

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String _copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| validationLevel | protected int validationLevel | The current validation level. |
| viewEvent | protected ViewEvent viewEvent | A reference to a ViewEvent. Create one ViewEvent reuse it between events. |
| data | protected Object data | A reference to the data. |

*FIG. 9A*

902

Constructors

| Name | Declaration | Description |
|---|---|---|
| ViewControllerImpl | public ViewControllerImpl() | Default constructor. |

*FIG. 9B*

ViewControllerImpl

904

Methods

| Name | Declaration | Description |
|---|---|---|
| addViewListener | public final void addViewListener (ViewListener listener) | Add a ViewListener. |
| clear | public void clear() | Clear local state by setting the data reference to null and by removing all ViewListeners. |
| exit | public void exit() | Get read to exit. Clear local state by setting the data reference to null, removing all Viewlisteners and setting view listeners to null. |
| fireViewEvent | public final void fireViewEvent(ViewEvent event) | If the ViewEvent is not null then send it to all ViewListeners |
| getComponent | public Component getComponent() | Return the Component that is "this" ViewController. By default, "this" is returned. Redefine this method in ViewControllerBaseImpl when you have a non-java.awt.Component superclass. |
| getJTCs | public Vector getJTCs() | Return all JTC type objects defined. By default null is returned. Typically, ViewControllers will not return anything. |
| getPermissions | public String[] getPermissions() | Return a set "keys" that can a management system can use when assigning JTC function based on roles (i.e. group, user). For example, consider the common case of operator override. In grocery store, if a cashier makes a mistake, a manager inserts a key or enters a password to enable more function on the cash register. The software analogy is that a button may become active or disabled. Suppose the ViewController implements a button labeled "Override" and it is the only component whose state can be visibly altered outside the ViewController. The ViewController writer will return: "Override" In this case, the only options are ENABLE or DISABLE. Suppose these constants are define to be 0x001 and 0x002, respectively. A management system that maintains user privileges is queried at runtime. The ViewController is then called with setPermissions(keys, values) where keys is "Override" and values is "0x001". The ViewController writer now responds to this request by turning off the button. Instead of hard coding the possible roles, the ViewController simply reacts to key/value settings. By default, nothing is returned. |

*FIG. 9C*

ViewControllerImpl (continued)

904

Methods

| Name | Declaration | Description |
|---|---|---|
| init | public void init() | Initialize, by default do nothing. |
| isEnabled | public boolean isEnabled() | Is this ViewController enabled? |
| isValid | public boolean isValid() | Is the ViewController in a consistent state? This usually means: Do all fields pass ValidationRules? The meaning could also be application specific. This value can provide other components with the ability to show a visual indicator, such as an X or a check in a tree menu indicating incomplete or partial data. The default value is true. |
| isVisible | public boolean isVisible() | Is this ViewController visible? |
| refresh | public void refresh(Object data) | Data objects are being passed in. By default, keep a reference to them. Interpretation of the data is performed in the subclass. For example, suppose the data being passed is a Customer object. Then a subclass can perform the following: This can be extended to more complex data types and data type composites (i.e. arrays, Vectors, etc.). |
| removeViewListener | public final void removeViewListener (ViewListener listener) | Remove a ViewListener. |
| setEnabled | public void setEnabled(boolean toggle) | Enable or disable the ViewController. Remember the state and ask the ViewControllerBaseImpl to handle it. |

*FIG. 9D*

ViewControllerImpl (continued)

*FIG. 9E*

904

Methods

| Name | Declaration | Description |
|---|---|---|
| setPermissions | public void setPermissions (Hashtable permissions) | Given a set of keys and values, update the internal state of the ViewController. The keys and values are supplied via a management system and relate to roles (i.e. users and groups). The possible values in the key/value pairs are application and ViewController specific. For example, create an interface to define the keys and values:<br>public interface Customer {<br>public static final String DETAILS="DETAILS";<br>public static final String ON="1";<br>public static final String OFF="0";<br>}<br>then set the ViewController:<br>Hashtable permissions=new Hashtable();<br>permissions.put(Customer.DETAILS, Customer.ON);<br>vc.setPermissions(Permissions);<br>The ViewController will interpret the meaning of ON and perform the necessary action, such as active a button. The meaning of keys, values and actions should be defined in a GUI spec. By default, nothing happens. |
| setProperties | public void setProperties(Properties properties) | Set the properties. Default is to do nothing. |
| setResources | public void setResources(ResourceBundle bundle) | Set the ResourceBundles. Default is to do nothing. |
| setValidationLevel | public void setValidationLevel(int level) | Set the validation level to indicate when ValidationRules should be applied<br>Four constants are defined in the ValidationRule class:<br>NONE<br>COMPONENT<br>FOCUS<br>VIEWEVENT<br>This value will be stored for the subclass to reference and act.<br>The default value is ValidationRule.NONE. |
| setVisible | public void setVisible(boolean visible) | Set the ViewController's visibility on or off.<br>Remember the state and ask the ViewControllerBaseImpl to handle it. |
| toString | public String toString() | Return the instance class name. |

ViewControllerBaseImpl

1000

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 10A

1002

Constructors

| Name | Declaration | Description |
|---|---|---|
| ViewControllerBaseImpl | public ViewControllerBaseImpl() | Default constructor. |

FIG. 10B

Methods

| Name | Declaration | Description |
|---|---|---|
| getComponent | public Component getComponent() | By default, return this. This works when the superclass is derived from java.awt.Component. Otherwise, override this method and return your own this, but be sure to override setEnabled and setVisible also. |
| setEnabled | public void setEnabled(boolean toggle) | By default, passes the call to the super class. |
| setVisible | public void setVisible(boolean visible) | By default, passes the call to the super class. |

FIG. 10C

ViewControllerAdapter

1100

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

*FIG. 11A*

1102

Constructors

| Name | Declaration | Description |
|---|---|---|
| ViewControllerAdapter | public ViewControllerAdapter() | Constructor. |

*FIG. 11B*

1104

Methods

| Name | Declaration | Description |
|---|---|---|
| actionPerformed | public void actionPerformed(ActionEvent e) | Do nothing. |
| adjustmentValueChanged | public void adjustmentValueChanged(AdjustmentEvent e) | Do nothing. |
| componentAdded | public void componentAdded(ContainerEvent e) | Do nothing. |
| componentHidden | public void componentHidden(ComponentEvent e) | Do nothing. |
| componentMoved | public void componentMoved(ComponentEvent e) | Do nothing. |
| componentRemoved | public void componentRemoved(ContainerEvent e) | Do nothing. |
| componentResized | public void componentResized(ComponentEvent e) | Do nothing. |
| componentShown | public void componentShown(ComponentEvent e) | Do nothing. |
| focusGained | public void focusGained(FocusEvent e) | Do nothing. |
| focusLost | public void focusLost(FocusEvent e) | Do nothing. |
| itemStateChanged | public void itemStateChanged(ItemEvent e) | Do nothing. |
| keyPressed | public void keyPressed(KeyEvent e) | Do nothing. |
| keyReleased | public void keyReleased(KeyEvent e) | Do nothing. |
| keyTyped | public void keyTyped(KeyEvent e) | Do nothing. |
| mouseClicked | public void mouseClicked(MouseEvent e) | Do nothing. |
| mouseDragged | public void mouseDragged(MouseEvent e) | Do nothing. |
| mouseEntered | public void mouseEntered(MouseEvent e) | Do nothing. |
| mouseExited | public void mouseExited(MouseEvent e) | Do nothing. |
| mouseMoved | public void mouseMoved(MouseEvent e) | Do nothing. |
| mousePressed | public void mousePressed(MouseEvent e) | Do nothing. |
| mouseReleased | public void mouseReleased(MouseEvent e) | Do nothing. |
| textValueChanged | public void textValueChanged(TextEvent e) | Do nothing. |

*FIG. 11C*

ValidationRule

*FIG. 12A*

1200

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| NONE | public static final int NONE | |
| COMPONENT | public static final int COMPONENT | |
| FOCUS | public static final int FOCUS | |
| VIEWEVENT | public static final int VIEWEVENT | |

*FIG. 12B*

1202

Constructors

| Name | Declaration | Description |
|---|---|---|
| ValidationRule | public ValidationRule() | |

*FIG. 12D*

1206

```
/**
* Given a list of class names, apply each validation rule of the classes
* to input string and return the formatted result.
*
* @return the viewable formatted string.
* @param classNames a comma-separated fully qualified list of concrete AbstractRule classes.
* @param input the input string to apply edit rules to.
* @exception ValidatonRuleException if there was an error in applying the edits.
*/
public static String applyEdits(String classNames, String input) throws ValidationRuleException {
      int commaIndex = -1;
      int curIndex = 0;
      do {
            commaIndex=classNames.indexOf(',', curIndex);
            if (commaIndex == -1) {
                  commaIndex = classNames.length();
            }
            String className = classNames.substring(curIndex, commaIndex).trim();
            try {
                  ValidationRule rule = (ValidationRule) Factory.newInstance(className);
                  input = rule.edit(input);
            } catch (ValidationRuleException re) {
                  throw re;
            } catch (Exception e) {
                  throw new ValidationRuleException("Rule class '" + className + "' not found.");
            }
            curIndex = commaIndex + 1;
      } while (curIndex < classNames.length());
      return input;
}
```

1204

Methods

| Name | Declaration | Description |
|---|---|---|
| applyEdits | public static String applyEdits (String classNames, String input) throws ValidationRuleException | Given a list of class names, apply each validation rule of the classes to input string and return the formatted result. Parameters: classNames – a comma-separated fully qualified list of concrete AbstractRule classes. input – the input string to apply edit rules to. Returns: the viewable formatted string. Throws: ValidationRuleException if there was an error in applying the edits. |
| applyNormalize | public static String applyNormalize (String classNames, String input) throws ValidationRuleException | Given a list of class names, apply each normalize rule of the classes to input string and return the transmittable result. Parameters: classNames – a comma-separated fully qualified list of concrete AbstractRule classes. input – the input string to apply normalize rules to. Returns: the transmittable string. Throws: ValidationRuleException |
| edit | public abstract String edit (String input) throws ValidationRuleException | Subclasses must implement this method to take an input string and apply some edit rule which returns a properly formatted string that can be used to display to the user. Parameters: input–the input string. Returns: the viewable formatted string. Throws: ValidationRuleException if unable to properly format input string. |
| normalize | public abstract String normalize (String input) throws ValidationRuleException | Subclasses must implement this method to take an input string and apply some normalize rule which returns a properly formatted string that can be used to send data to some server. Parameters: input – the input string. Returns: the transmittable string. Throws: ValidationRuleException if unable to properly format input string. |

*FIG. 12C*

ValidationRuleException

1300

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

*FIG. 13A*

1302

Constructors

| Name | Declaration | Description |
|---|---|---|
| ValidationRuleException | public ValidationRuleException() | Default constructor. |
| ValidationRuleException | public ValidationRuleException(String s) | Constructor with a message to the rule exception. |

*FIG. 13B*

ViewEvent

*FIG. 14A*

1400

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| VIEWEVENT_FIRST | public static final int VIEWEVENT_FIRST | |
| OK | public static final int OK | |
| DONE | public static final int DONE | |
| OPEN | public static final int OPEN | |
| CLOSE | public static final int CLOSE | |
| CANCEL | public static final int CANCEL | |
| EXIT | public static final int EXIT | |
| FILE | public static final int FILE | |
| SAVE | public static final int SAVE | |
| SAVEAS | public static final int SAVEAS | |
| ERROR | public static final int ERROR | |
| WARNING | public static final int WARNING | |
| RETURN | public static final int RETURN | |
| LOAD | public static final int LOAD | |
| NOTIFY | public static final int NOTIFY | |
| NOTIFY2 | public static final int NOTIFY2 | |
| INFO | public static final int INFO | |
| SETUP | public static final int SETUP | |
| PRINT | public static final int PRINT | |

ViewEvent
(continued)

1400

Variables

| Name | Declaration | Description |
|---|---|---|
| TITLEMESSAGE | public static final int TITLEMESSAGE | |
| STATUSMESSAGE | public static final int STATUSMESSAGE | |
| ERRORMESSAGE | public static final int ERRORMESSAGE | |
| SUGGESTIONMESSAGE | public static final int SUGGESTIONMESSAGE | |
| NEXT | public static final int NEXT | |
| PREVIOUS | public static final int PREVIOUS | |
| FIRST | public static final int FIRST | |
| LAST | public static final int LAST | |
| START | public static final int START | |
| BEGIN | public static final int BEGIN | |
| END | public static final int END | |
| PAUSE | public static final int PAUSE | |
| STOP | public static final int STOP | |
| RESTART | public static final int RESTART | |
| SUBMIT | public static final int SUBMIT | |
| BACKSPACE | public static final int BACKSPACE | |
| INSERT | public static final int INSERT | |

*FIG. 14B*

ViewEvent (continued)

FIG. 14C

1400

Variables

| Name | Declaration | Description |
|---|---|---|
| HOME | public static final int HOME | |
| PGUP | public static final int PGUP | |
| PGDN | public static final int PGDN | |
| LEFT | public static final int LEFT | |
| RIGHT | public static final int RIGHT | |
| UP | public static final int UP | |
| DOWN | public static final int DOWN | |
| LIST | public static final int LIST | |
| MORE | public static final int MORE | |
| ADD | public static final int ADD | |
| DELETE | public static final int DELETE | |
| MODIFY | public static final int MODIFY | |
| NEW | public static final int NEW | |
| EDIT | public static final int EDIT | |
| COPY | public static final int COPY | |
| CUT | public static final int CUT | |
| PASTE | public static final int PASTE | |
| UNDO | public static final int UNDO | |
| REMOVE | public static final int REMOVE | |
| PLUS | public static final int PLUS | |
| MINUS | public static final int MINUS | |
| INCREMENT | public static final int INCREMENT | |
| DECREMENT | public static final int DECREMENT | |
| CHANGED | public static final int CHANGED | |

ViewEvent (continued)

FIG. 14D

1400

Variables

| Name | Declaration | Description |
|---|---|---|
| FILL | public static final int FILL | |
| EMPTY | public static final int EMPTY | |
| READY | public static final int READY | |
| VIEW | public static final int VIEW | |
| DETAILS | public static final int DETAILS | |
| READ | public static final int READ | |
| WRITE | public static final int WRITE | |
| SEARCH | public static final int SEARCH | |
| FIND | public static final int FIND | |
| HELP | public static final int HELP | |
| HINT | public static final int HINT | |
| TRAIN | public static final int TRAIN | |
| TEACH | public static final int TEACH | |
| SUGGEST | public static final int SUGGEST | |
| VIEWEVENTTEST1 | public static final int VIEWEVENTTEST1 | |
| VIEWEVENTTEST2 | public static final int VIEWEVENTTEST2 | |
| VIEWEVENTTEST3 | public static final int VIEWEVENTTEST3 | |
| VIEWEVENT_LAST | public static final int VIEWEVENT_LAST | |
| consumed | protected boolean consumed | Is event still valid? |
| timestamp | protected long timestamp | Time stamp when event is fired. |
| data | protected Object data | Data reference. |

FIG. 14E

1402

Constructors

| Name | Declaration | Description |
|---|---|---|
| ViewEvent | public ViewEvent() | Constructs a ViewEvent. |
| ViewEvent | public ViewEvent(Object source) | Constructs a ViewEvent. |
| ViewEvent | public ViewEvent(Object source, int major) | Constructs a ViewEvent object with the specified source object and code; |
| ViewEvent | public ViewEvent(Object source, int major, int mirror, Object data) | Constructs a ViewEvent object with the specified source object and code; |
| ViewEvent | public ViewEvent(Object source, int major, Object data) | Constructs a ViewEvent object with the specified source object and code; |

ViewEvent (continued)

FIG. 14F

1404

Methods

| Name | Declaration | Description |
|---|---|---|
| consume | public final void consume() | Consume this event. |
| getData | public Object getData() | Return the data. |
| getMajor | public final int getMajor() | Return the major event code. |
| getMinor | public final int getMinor() | Return the event option. |
| getSource | public final Object getSource() | Gets the event source Overrides: getSource in class EventObject. |
| getTimestamp | public long getTimestamp() | Get the timestamp when the event was fired. By default, this was set by JTC. |
| isConsumed | public final boolean isConsumed() | Is the event consumed? |
| setConsumed | public final void setConsumed(boolean consumed) | Turn event consumed on/off. |
| setData | public void setData(Object data) | Sets the data. |
| setMajor | public final void setMajor(int code) | Sets the event code. |
| setMinor | public final void setMinor(int code) | Sets the event option. |
| setSource | public final void setSource(Object source) | Sets the event source. |
| setTimestamp | public void setTimestamp(long time) | Set the timestamp when the event is fired. By default, this is set by JTC. |
| toString | public String toString() | Returns a string representation of the object.<br>The class of the event and the reason for the event is returned. |

ViewListener

1500

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

*FIG. 15A*

1502

Methods

| Name | Declaration | Description |
|---|---|---|
| viewEventPerformed | public abstract void viewEventPerformed(ViewEvent event) | Invoked when a ViewEvent has been fired. |

*FIG. 15B*

ApplicationMediator

1600

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

*FIG. 16A*

FIG. 16B

1602

Methods

| Name | Declaration | Description |
|---|---|---|
| addPlacementListener | public abstract void addPlacementListener (PlacementListener listener) | Invoked when a PlacementListener is added. |
| addRequestListener | public abstract void addRequestListener (RequestListener listener) | |
| addTopListener | public final void addTopListener (TopListener listener) | Add a TopListener. |
| addViewListener | public abstract void addViewListener (ViewListener listener) | Invoked when a ViewListener is added. |
| getPermissions | public abstract String[] getPermissions() | Invoked when the ApplicationMediator permission keys are needed. |
| init | public abstract void init (ApplicationMediator applicationMediator) | Invoked when an ApplicationMediator should be initialized based on another ApplicationMediator's contents. |
| isValid | public abstract boolean isValid() | Invoked when the ApplicationMediator's state needs to be checked to see if it is valid. |
| isVisible | public abstract boolean isVisible() | Invoked to see if the ApplicationMediator is visible. |
| refresh | public abstract void refresh (Object data) | Invoked to supply new or changed data. |
| removePlacementListener | public abstract void removePlacementListener (PlacementListener listener) | Invoked to remove a PlacementListener. |
| removeRequestListener | public abstract void removeRequestListener (RequestListener listener) | Invoked to remove a RequestListener. |
| removeTopListener | public final void removeTopListener (TopListener listener) | Removes the TopListener. |
| removeViewListener | public abstract void removeViewListener (ViewListener listener) | Invoked to remove a ViewListener. |
| setPermissions | public abstract void setPermissions (Hashtable permissions) | Invoked to set the permissions keys and values. |
| setProperties | public abstract void setProperties (Properties properties) | Invoked to set the properties. |
| setResouces | public abstract void setResources (ResourceBundle bundle) | Invoked to set the resources. |
| setVisible | public abstract void setVisible(boolean visible) | Invoked to set the visibility. |

ApplicationMediatorImpl

1700

Variables

| Name | Declaration | Description |
|---|---|---|
| placementListeners | protected Vector placementListeners | The PlacementListeners. |
| topListeners | protected TopListener topListener | The TopListener. |
| requestListeners | protected Vector requestListeners | The RequestListeners. |
| viewListeners | protected Vector viewListeners | The ViewEventListeners. |
| viewControllers | protected Vector viewControllers | Whenever view controllers are created, it is by convention they will be added to this array. |
| applicationMediators | protected Vector applicationMediators | Whenever application mediators are created, it is by convention they will be added to this array. |
| data | protected Object data | This is a reference to the system data model. |
| requestEvent | protected RequestEvent requestEvent | This is a reference to a RequestEvent. |

*FIG. 17A*

1702

Constructors

| Name | Declaration | Description |
|---|---|---|
| ApplicationMediatorImpl | public ApplicationMediatorImpl() | Constructor. By changing commented code, you can switch between threading styles 1 and 2. |

*FIG. 17B*

ApplicationMediatorImpl

*FIG. 17C*

1704

Methods

| Name | Declaration | Description |
|---|---|---|
| addPlacementListener | public final void addPlacementListener(PlacementListener listener) | Add a PlacementListener. |
| addRequestListener | public final void addRequestListener(RequestListener listener) | Add a RequestListener. |
| addTopListener | public final void addTopListener(TopListener listener) | Add a TopListener. |
| addViewListener | public final void addViewListener(ViewListener listener) | Add a ViewListener. |
| clear | public void clear() | Clear the ApplicationMediator by clearing all allocated ViewControllers and ApplicationMediators. All data is set to null, but lists are not destroyed. A 'cleared' ApplicationMediator can be used again. If this method is overriden in a subclass, be sure to invoke super.clear(); |
| exit | public void exit() | Exit the ApplicationMediator by exiting all allocated ViewControllers and ApplicationMediators. All data is set to null, and lists are destroyed. An 'exited' ApplicationMediator cannot be used again. If this method is overriden in a subclass, be sure to invoke super.exit(); |
| firePlacementEvent | protected final void firePlacementEvent(PlacementEvent event) | Notify the PlacementListeners. |
| fireRequestEvent | protected final void fireRequestEvent(RequestEvent event) throws RequestException | Notify the RequestListeners – synchronous. |
| fireRequestEvent | protected final void fireRequestEvent(RequestEvent event, RequestResponseListener caller) throws RequestException | Notify the RequestListeners – asynchronous. |
| fireTopEvent | protected final void fireTopEvent(TopEvent event) | Notify the TopListeners. |
| fireTopListener | | |
| fireViewEvent | protected final void fireViewEvent(ViewEvent event) | Notify the ViewListeners. |
| getAM | protected ApplicationMediator getAM(int i) | Return the i'th ApplicationMediator. |
| getJTCs | public Vector getJTCs() | Return a vector of all ThinClient objects. By default, this is a Vector containing the created ViewControllers and ApplicationMediators. |

ApplicationMediatorImpl (continued) *FIG. 17D* 1704

Methods

| Name | Declaration | Description |
|---|---|---|
| getPermissions | public String[] getPermissions() | Get the settable permission keys. By default, return the class names of all allocated ViewControllers and ApplicationMediators. |
| getVC | protected ViewController getVC(int i) | Return the i'th ViewController |
| init | public void init() | Initialize the ApplicationMediator, nothing to do by default. |
| init | public void init(ApplicationMediator applicationMediator) | Initialize the ApplicationMediator using the listeners of an existing ApplicationMediator. |
| initApplicationMediators | public final void initApplicationMediators(String classnames[]) throws ClassNotFoundException, InstantiationException, IllegalAccessException | For each ApplicationMediator classname, load it, new it and add myself as a ViewEvent. The Factory class is used as helper class. |
| initViewControllers | public final void initViewControllers(String classnames[]) throws ClassNotFoundException, InstantiationException, IllegalAccessException | For each ViewController classname, load it, new it and add myself as a ViewEvent. The Factory class is used as helper class. |
| isEnabled | public boolean isEnabled() | Is the ApplicationController enabled? |
| isValid | public boolean isValid() | Return the AND'ed value of calling isValid on ApplicationMediators and ViewControllers. |
| isVisible | public boolean isVisible() | Is the ApplicationController visible? Hardly, since it is a non visible class. But this looks to see if any of its ViewControllers are visible. Not really, they were all set to visible/invisible via the setVisible method and we remembered the state to return here. |
| processViewEvent | public abstract void processViewEvent(ViewEvent e) | Deliver the ViewEvent to the subclass via this method. |
| refresh | public void refresh(Oject data) | When new data arrives allow the ViewControllers and ApplicationControllers to be refreshed also. |
| removePlacementListener | public final void removePlacementListener(PlacementListener listener) | Removes the PlacementListener. |

FIG. 17E

1704

Methods

| Name | Declaration | Description |
|---|---|---|
| removeRequestListener | public final void removeRequestListener(RequestListener listener) | Removes the RequestListener. |
| removeViewListener | public final void removeViewListener(ViewListener listener) | Removes the ViewListener. |
| requestException | public void requestException(RequestException yikes) | Called back because an asynchronous request has thrown an Exception. By default, print the message to System.err. |
| requestResponse | public void requestResponse(RequestEvent response) | Called back with the results of an asynchronous request. By default, call refresh with the data in the response. |
| run2 | public final void run2() | This method is used in style 1 threading. Rename this to run() and uncomment the code as described in the class javadoc. |
| setAM | public void setAM(ApplicationMediator applicationMediator, int i) | Set the i'th ApplicationMediator. |
| setEnabled | public void setEnabled(boolean toggle) | Call setEnabled on each ViewController and ApplicationMediator. |
| setPermissions | public void setPermissions(Hashtable permissions) | Set the permissions. By default, call setPermissions on each ViewController and ApplicationMediator. |
| setProperties | public void setProperties(Properties properties) | Set the properties. By default, call setProperties on each ViewController and ApplicationMediator. |
| setResources | public void setResources(ResourceBundle bundle) | Set the resources. By default, call setResources on each ViewController and ApplicationMediator. |
| setVC | public void setVC(ViewController viewController, int i) | Set the i'th ViewController. |
| setVisible | public void setVisible(boolean visible) | Set visible on each ViewController and ApplicationMediator. |
| toString | public String toString() | Return the Class name of the ApplicationController instance. |
| viewEventPerformed | public void viewEventPerformed(ViewEvent e) | A ViewEvent is delivered. Process it using Threading style 1 or 2. In the end, the processViewEvent will be called on the subclass. |

1706

ApplicationMediator Impl.exit(): AUS8-1999-0694

```
/**
 * Exit the ApplicationMediator by exiting all allocated ViewControllers
 * and ApplicationMediators. All data is set to null, and lists are
 * destroyed. An 'exited' ApplicationMediator cannot be used again.
 * If this method is overriden in a subclass, be sure to invoke
 * super.exit();
 **/
public void exit() {
     synchronized (this) {

          /* Used for style 1 event dispatching. Leave this code commented. */
          //if (this.eventThread !=null) {
          //    try {
          //         eventThread.stop ();
          //    } catch (Exception e) {
          //    }
          //}

          /* Used for style 2 event dispatching. Leave this code commented. */
          for (int i = 0; i < runningThreads.size(); i++) {
               ((ApplicationMediatorThread) runningThreads.elementAt (i)) .stop();
          }
          runningThreads.removeAllElements();
          viewListeners.removeAllElements();
          try {
               for (int i = 0; i < viewControllers.size(); i++) {
                    ((ViewController) viewControllers.elementAt(i)) .setEnabled(false);
                    ((ViewController) viewControllers.elementAt(i)) .exit ();
               }
               for (int i = 0; i < applicationMediators.size(); i++) {
                    ((ApplicationMediator) applicationMediators.elementAt(i)) .setEnabled(false);
                    ((ApplicationMediator) applicationMediators.elementAt(i)) .exit();
               }
          } catch (Exception noProblem) {
          }
          viewControllers = null;
          applicationMediators = null;
          runningThreads = null;
          runningThreads = null;
          data = null;
     }
}
```

*FIG. 17F*

1708

ApplicationMediatorImpl.clear(): AUS8-1999-0694

```
/**
 * Clear the ApplicationMediator by clearing all allocated ViewControllers
 * and ApplicationMediators. All data is set to null, but lists are
 * not destroyed. A 'cleared' ApplicationMediator can be used again.
 * If this methed is overriden in a subclass, be sure to invoke
 * super.clear();
 */
public void clear() {
     synchronized (this) {

          /* Used for style 1 event dispatching. Leave this code commented. */
          //if (this.eventThread != null) {
          //    try {
          //         eventThread.stop ();
          //    } catch (Exception e) {
          //    }
          //}

          /* Used for style 2 event dispatching. Leave this code commented. */
          for (int i = 0; i < runningThreads.size(); i++) {
               ((ApplicationMediatorThread) runningThreads.elementAt (i)) .stop();
          }
          runningThreads.removeAllElements();

          //
          try {
               for (int i = 0; i < viewControllers.size(); i++) {
                    ((ViewController) viewControllers.elementAt(i)) .setEnabled(false);
                    ((ViewController) viewControllers.elementAt(i)) .clear ();
               }
               for (int i = 0; i < applicationMediators.size(); i++) {
                    ((ApplicationMediator) applicationMediators.elementAt(i)) .setEnabled(false);
                    ((ApplicationMediator) applicationMediators.elementAt(i)) .clear();
               }
          } catch (Exception noRealProblem) {
          }
          viewControllers = null;
          applicationMediators = null;
          data = null;
          viewListeners.removeAllElements();
     }
}
```

*FIG. 17G*

1710

```
/**
 * Initalize the ApplicationMediator using the listeners of an
 * existing ApplicationMediator.
 */
public void init(ApplicationMediator applicationMediator) {
      if (applicationMediator instanceof ApplicationMediatorImpl) {
            ApplicationMediatorImpl a = (ApplicationMediatorImpl) applicationMediator;
            requestListeners = (Vector) a.requestListeners.clone();
            placementListeners = (Vector) a.placementListeners.clone();
            topListeners = (Vector) a.topListeners.clone();
            addViewListener(a);
      }
      init();
}
```

*FIG. 17H*

1712

```
/**
 * When new data arrives allow the ViewControllers
 * and ApplicationControllers to be refreshed also.
 */
public void refresh(Object data) {
      this.data = data;
      try {
            synchronized (viewControllers) {
                  for (int j = 0; j < viewControllers.size(); j++) {
                        ((ViewController) viewControllers.elementAt(j)).
                              refresh(data);
                  }
            }
      } catch (Exception noRealProblem) {
      }
      try {
            synchronized (applicationMediators) {
                  for (int j = 0; j < applicationMediators.size(); j++) {
                        ((ApplicationMediator) applicationMediators.
                              elementAt(j)).refresh(data);
                  }
            }
      } catch (Exception noRealProblem) {
      }
}
```

*FIG. 17I*

1714

```
/**
 * A ViewEvent is delivered. Process it using Threading style 1 or 2. In
 * the end, the processViewEvent will be called on the subclass.
 */
public void viewEventPerformed (ViewEvent e) {
      /* Used for style 2 event dispatching, start an inner class thread */
      ApplicationMediatorThread t = new ApplicationMediatorThread (e);
      runningThreads.addElement (t);
      t.start ();

      /* Used for style 1 event dispatching. Leave this code commented. */
      //ViewEvent saved = saveViewEvent(e);
      //if (eventThread == null || !eventThread.isAlive()) {
      // finished = false;
      // eventThread = new Thread(this);
      // eventThread.start ();
      //}
      //synchronized (this) {
      // notify();
      //}
}
```

*FIG. 17J*

1714

```
/**
 * This method is used in style 1 threading. Rename this to run ()
 * and uncomment the code as described in the class javadoc.
 */
public final void run2 () {
    /* Used for style 1 event dispatching. Leave this code commented. */
    /*
    while (true) {
            ViewEvent event = null;
        event = getViewEvent ();
        if (event != null) {
            handleViewEvent (event);
        } else {
            waitForEvent ();
            if (finished) {
            // something went wrong with the thread so hose this loop
            break;
            }
        }
    }
    */
}
```

*FIG. 17K*

1714

```
/**
* Private class to handle executions of ViewEvents () on another thread.
**/
private class ApplicationMediatorThread extends Thread {
    /**
    * The current event
    **/
    private ViewEvent event;
    /**
    * Create an ApplicationMediatorThread to process the ViewEvent
    **/
    public ApplicationMediatorThread(ViewEvent event) {
        super ();
        this.event = event;
    }
    /**
    * Just call the handleViewEvent method that the subclass will override
    **/
    public void run () {
        processViewEvent (event);
    }
}
```

*FIG. 17L*

1714

```
/**
 * Save the current ViewEvent on a Q
 */
private final ViewEvent saveViewEvent (ViewEvent e) {
    /* Used for style 1 event dispatching. Leave this code commented. */
    //return viewEventQueue.add(e);
    return null;
}

/**
 * Method: return the first view event saved. Used by the Q'ing system.
 */
private ViewEvent getViewEvent () {
    /* Used for style 1 event dispatching. Leave this code commented. */
    //return (ViewEvent) viewEventQueue.remove();
    return null;
}
```

*FIG. 17M*

PlacementEvent

FIG. 18A

1800

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| PLACEMENTEVENT_FIRST | public static final int PLACEMENTEVENT_FIRST | |
| ADD | public static final int ADD | |
| REMOVE | public static final int REMOVE | |
| MODIFY | public static final int MODIFY | |
| PLACEMENTEVENT_LAST | public static final int PLACEMENTEVENT_LAST | |
| major | protected int major | The placementevent code |
| minor | protected int minor | The placementevent option |
| component | protected Object component | Component Reference |
| data | protected Object data | Data reference |



FIG. 18B

1802

Constructors

| Name | Declaration | Description |
|---|---|---|
| PlacementEvent | public PlacementEvent() | Constructs a PlacementEvent |
| PlacementEvent | public PlacementEvent(Object source, Object component) | Constructs a PlacementEvent |
| PlacementEvent | public PlacementEvent(Object source, Object component, int major) | Constructs a PlacementEvent |
| PlacementEvent | public PlacementEvent(Object source, Object component, int major, int minor) | Constructs a PlacementEvent |
| PlacementEvent | public PlacementEvent(Object source, Object component, int major, int minor, Object data) | Constructs a PlacementEvent |

FIG. 18C

1804

Methods

| Name | Declaration | Description |
|---|---|---|
| getComponent | public final Component getComponent() | Return the Component |
| getData | public final Object getData() | Return the data |
| getMajor | public final int getMajor() | Return the major code |
| getMinor | public final int getMinor() | Return the minor code |
| getSource | public final Object getSource() | Gets the event source |
| setComponent | public final void setComponent(Component component) | Sets the Component |
| setData | public final void setData(Object data) | Set the data |
| setMajor | public final void setMajor(int code) | Set the major code |
| setMinor | public final void setMinor(int code) | Sets the minor code |
| setSource | public final void setSource(Object source) | Set the event source |
| toString | public String toString() | Returns a string representation of the object. |

PlacementListener

FIG. 19A

1900

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 19B

1902

Methods

| Name | Declaration | Description |
|---|---|---|
| placementEventPerformed | public abstract void placementEventPerformed (PlacementEvent event) | Invoked when we are being called to add/remove/modify a component. Do it. |

TopEvent

*FIG. 20A*

2000

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| TOPEVENT_FIRST | public static final int TOPEVENT_FIRST | |
| EXIT | public static final int EXIT | |
| BROWSER | public static final int BROWSER | |
| TITLE | public static final int TITLE | |
| STATUS | public static final int STATUS | |
| OS | public static final int OS | |
| A | public static final int A | |
| B | public static final int B | |
| C | public static final int C | |
| D | public static final int D | |
| E | public static final int E | |
| F | public static final int F | |
| TRACE | public static final int TRACE | |
| DEBUG | public static final int DEBUG | |
| LOG | public static final int LOG | |
| HOOKAWT | public static final int HOOKAWT | |
| HOOKJTC | public static final int HOOKJTC | |
| TOPEVENT_LAST | public static final int TOPEVENT_LAST | |
| TEAM | public static final int TEAM | |
| WIN | public static final int WIN | |
| EXECUTE | public static final int execute | |
| consumed | protected boolean consumed | Is event still valid? |
| data | protected Object data | This is a loose reference to the data model. We don't care what the class shape is and we only reference it via the interface that it must implement. |

TopEvent

FIG. 20B

2002

Constructors

| Name | Declaration | Description |
|---|---|---|
| TopEvent() | public TopEvent () | Default constructor for a Request. |
| TopEvent(Object) | public TopEvent (Object source) | Construct with the given source and default major and minor values. |
| TopEvent(Object, int) | public TopEvent (Object source, int major) | Create a Request with a source, major and minor codes. |
| TopEvent(Object, int, int) | public TopEvent (Object source, int major, int minor) | Create a Request with major and minor codes. |
| TopEvent(Object, int, int, Object) | public TopEvent (Object source, int major, int minor, Object data) | Create a Request with a source, major and minor codes, and some data. If source is null, an InvalidArgumentException will be thrown. |

FIG. 20C

2004

Methods

| Name | Declaration | Description |
|---|---|---|
| consume | public final void consume () | Consume this event. |
| getData | public final Object getData () | Return the reference to the data. |
| getMajor | public final int getMajor () | Get the major code. |
| getMinor | public final int getMinor () | Get the minor code. |
| getSource | public final Object getSource () | Gets the event source. Overrides: getSource in class EventObject. |
| isConsumed | public final boolean isConsumed () | Is the event consumed? |
| setConsumed | public final void setConsumed (boolean consumed) | Turn event consumed on or off. |
| setData | public final void setData (Object data) | Set the data. |
| setMajor | public final void setMajor (int major) | Set the major code. |
| setMinor | public final void setMinor (int minor) | Set the minor code. This is always a String. |
| setSource | public final void setSource (Object source) | Sets the event source. |
| toString | public String toString () | Show a String representation of the Request in the format of "TopEvent(major,minor)". |

TopListener

2100

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

*FIG. 21A*

2102

Methods

| Name | Declaration | Description |
|---|---|---|
| exec | public abstract void exec (Object programInformation) | Invoked to execute a desktop program. The parameter programInformation can be a complex object with lots of data. For example: String[] params = {"netscape.exe", "http://www.ibm.com"}; aTopListener.exec(params). Another usage might be to interact with JavaScript under a browser. It is up to the TopListener implementer to understand what the params mean. Do not create a language with a language. This method should only be defined to support legacy environments or corporate desktop rules. Consider using a RequestEvent for more complex requirements. |
| exit | public abstract void exit() | Invoked to exit a JTC application. Never let a program perform its own "exit". This shuts the JVM down. The implementer of TopListener will know the appropriate actions to take to exit from an application on a corporate desktop. |
| message | public abstract void message (Object messageInfo) | Invoked to show a business specific message. Try to isolate calls to the browser here. |
| title | public abstract void title (Object titleInfo) | Invoked to display a business specific title. Try to isolate calls to a browser or a desktop program to display titles here. |
| topEventPerformed | public abstract void topEventPerformed (TopEvent event) | Invoked when we are being called to perform a top desktop function. |

*FIG. 21B*

RequestEvent

2200

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| consumed | protected boolean consumed | Is event still valid? |
| data | protected Object data | This is a loose reference to the data model. We don't care what the class shape is and we only reference it via the interface that it must implement. |

*FIG. 22A*

2202

Constructors

| Name | Declaration | Description |
|---|---|---|
| RequestEvent | public RequestEvent() | Default constructor for a Request. |
| RequestEvent | public RequestEvent(Object source) | Construct with the given source and default major and minor values. |
| RequestEvent | public RequestEvent(Object source, String major) | Create a Request with a source, major and minor codes. |
| RequestEvent | public RequestEvent(Object source, String major, String minor) | Create a Request with major and minor codes. |
| RequestEvent | public RequestEvent(Object source, String majorCode, String minorCode, Object data) | Create a Request with a source, major and minor codes, and some data. If source is null, an InvalidArgumentException will be thrown. |

*FIG. 22B*

2204

Methods

| Name | Declaration | Description |
|---|---|---|
| consume | public final void consume() | Consume this event. |
| getData | public final Object getData() | Return the reference to the data. |
| getMajor | public final String getMajor() | Get the major code. This is always a String. |
| getMinor | public final String getMinor() | Get the minor code. This is always a String. |
| getSource | public final Object getSource() | Gets the event source. |
| getStatus | public final String getStatus() | Return the status. |
| isConsumed | public final boolean isConsumed() | Is the event consumed? |
| setConsumed | public final void setConsumed(boolean consumed) | Turn event consumed on or off. |
| setData | public final void setData(Object data) | Set the data. |
| setMajor | public final void setMajor(String major) | Set the major code. This is always a String. |
| setMinor | public final void setMinor(String minor) | Set the minor code. This is always a String. |
| setSource | public final void setSource(Object source) | Sets the event source. |
| setStatus | public final void setStatus(String message) | Append a message to the status. |
| toString | public String toString() | Show a String representation of the Request in the format of "RequestEvent(major,minor)". |

FIG. 22C

RequestException

2300

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 23A

2302

Constructors

| Name | Declaration | Description |
|---|---|---|
| RequestException | public RequestException() | Default constructor. |
| RequestException | public RequestException(String s) | Constructor with a message to the request exception. |
| RequestException | public RequestException(Throwable target) | Constructor with a throwable target. |
| RequestException | public RequestException(Throwable target, String s) | Constructor with a throwable target and a message. |

FIG. 23B

2304

Methods

| Name | Declaration | Description |
|---|---|---|
| getTargetException | public Throwable getTargetException() | Get the target throwable. |
| setTargetException | public void setTargetException(Throwable target) | Set the target throwable. |
| toString | public String toString() | String version. |

FIG. 23C

RequestListener

FIG. 24A

2400

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 24B

2402

Methods

| Name | Declaration | Description |
|---|---|---|
| requestEventPerformed | public abstract void requestEventPerformed (RequestEvent request) throws RequestException | Invoked for a synchronous RequestEvent. |
| requestEventPerformed | public abstract void requestEventPerformed (RequestEvent request, RequestResponseListener listener) throws RequestException | Invoked for an asynchronous RequestEvent. |

RequestResponseListener

FIG. 25A

2500

Variables

| Name | Declaration | Description |
|---|---|---|
| copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 25B

2502

Methods

| Name | Declaration | Description |
|---|---|---|
| requestException | public abstract void requestException (RequestException yikes) | Invoked when an exception occured during processing of an asynchronous RequestEvent. |
| requestResponse | public abstract void requestResponse (RequestEvent result) | Invoked when the processing of an asynchronous RequestEvent was successful. |

Transporter

2600

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| PRIORITY | public static final String PRIORITY | Priority symbol. |
| WILDCARD | public static final String WILDCARD | Wildcard symbol. |

*FIG. 26A*



2602

Constructors

| Name | Declaration | Description |
|---|---|---|
| Transporter | public Transporter() | Default constructor. |

*FIG. 26B*

Transporter

FIG. 26C

2604

Methods

| Name | Declaration | Description |
|---|---|---|
| addDestinationListener | public void addDestinationListener (Object major, Destination destination) | Add the Destination using the given major code. If the destination is present with the same major don't re-add it - only one major/destination pair can exist. If the major is present, but the destination isn't, add the destination to the list of other destinations with the same key. If the key isn't present, store it and then add the new destination. If the destination is disabled, do nothing. |
| clear | public void clear() | For each RequestEvent not started, a RequestException will be thrown and the internal data structures will be emptied including RequestEvent queues and listeners. |
| exit | public void exit() | For each RequestEvent not started, a RequestException will be thrown and the internal data structures will be emptied including RequestEvent queues and listeners. All variable references will be set to null. |
| getDestinations | public synchronized Vector getDestinations() | Return a Vector of all Destinations currently registered. |
| getDestinations | public Vector getDestinations(Object major) | Return a Vector of the Destinations currently registered for the given major code. |
| getJTCs | public Vector getJTCs() | Return allocated JTC objects. By default, return the Destinations. |
| getMajorCodes | public Vector getMajorCodes() | Return a Vector of the registered major codes. |
| init | public void init() | Initialize the transporter. By default, do nothing. |
| isEnabled | public boolean isEnabled() | Is this Transporter enabled or disabled? A Transporter that is disabled will not process a RequestEvents but will throw RequestExceptions. |

Transporter (continued)

FIG. 26D

2604

Methods

| Name | Declaration | Description |
|---|---|---|
| isTagging | public boolean isTagging() | Is this Transporter tagging RequestEvents? |
| processDestinations | protected void processDestinations(RequestEvent request, Vector currentDestinations) throws RequestException | Given a RequestEvent and a Vector of destinations, call each Destination in FIFO/FEFR order. If tagging is enabled, then append a status tag to the RequestEvent. |
| removeDestinationListener | public void removeDestinationListener (Object major, Destination d) | Remove the destination using the given major. If the destination is not present, do nothing. If the destination is present, just remove it. If it was the last destination, remove all references to the major code. |
| requestEventPerformed | public void requestEventPerformed(RequestEvent request) throws RequestException | Submit a synchronous request. For each Destination that is listening for the current family of RequestEvents (the major code), send the RequestEvent to the Destination for processing. If there is a problem, throw a RequestException. Continue processing the RequestEvent as long as a RequestException is not thrown by a Destination and the RequestEvent is not consumed. If tagging is enabled, then append a status tag to the RequestEvent. Destinations are processed in the following FIFO order: 1-All using "!" (priority). 2-All using a major code. 3-All using "*". |
| requestEventPerformed | public void requestEventPerformed(RequestEvent request, RequestResponseListener caller) throws RequestException | Submit an asynchronous request. See the synchronous requestEventPerformed for more information. |
| setEnabled | public void setEnabled(boolean toggle) | Enable or disable the Transporter. A disabled Transporter will throw RequestExceptions if accessed via requestEventPerformed. |
| setRequestTagging | public void setRequestTagging(boolean toggle) | Stop or start the tagging of Requests. |
| toString | public String toString() | Return the String Transporter plus the number of registered Destinations. |

2604

Transporter.processDestinations(RequestEvent, Vector):AUS8-1999-0693

```
/**
 * Given a RequestEvent and a Vector of destinations, call each Destination
 * in FIFO/FEFR order.
 * <p>
 * If tagging is enabled, then append a status tag to the RequestEvent.
 * @exception RequestException if the Request can't be submitted
 */
protected void processDestinations(RequestEvent request, Vector currentDestinations) throws RequestException {
    if (!enabled) {
        throw new RequestException("Transporter disabled");
    }
    if (currentDestinations == null)
        return;

    /* process FIFO/FEFR */
    Destination d = null;
    int size = currentDestinations.size();
    for (int i = 0; !request.isConsumed() && i < size; i++) {
        d = (Destination) currentDestinations.elementAt(i);
        d.requestEventPerformed(request);
        /* Try to tag the request */
        if (tagging)
            request.setStatus (request.getStatus() + d);
    }
}
```

*FIG. 26E*

Transporter.requestEventPerformed(RequestEvent):AUS8-1999-0693

```
/**
 * Submit a synchronous request. For each Destination that is listening for
 * the current family of RequestEvents (the major code), send the RequestEvent
 * to the Destination for processing. If there is a problem, throw
 * a RequestException. Continue processing the RequestEvent as long
 * as a RequestException is not thrown by a Destination and the RequestEvent
 * is not consumed.
 * <p>
 * If tagging is enabled, then append a status tag to the RequestEvent.
 *<p>
 * Destinations are processed in the following FIFO order:
 * 1- All using "!" (priority).
 * 2- All using a major code.
 * 3- All using "*".
 *<p>
 * @exception RequestException if the Request can't be submitted
 */
public void requestEventPerformed(RequestEvent request) throws RequestException {
     if (!enabled) {
          throw new RequestException("Transporter disabled");
     }

     /* Try to tag the request */                                    2606
     if (tagging)
          request.setStatus(request.getStatus() + "[Transporter]");
     /* Process PRIORITY, major and then WILDCARD destinations */
     processDestinations(request, getDestinations(PRIORITY));
     processDestinations(request, getDestinations(request.getMajor()));
     processDestinations(request, getDestinations(WILDCARD));
}
```

*FIG. 26F*

```
/**
 * Submit an asynchronous request. See the synchronous
 * requestEventPerformed for more information.
 */
public void requestEventPerformed(RequestEvent request,
RequestResponseListener caller) throws RequestException {
     if (!enabled) {
          throw new RequestException("Transporter disabled");
     }
     if (tagging)
          request.setStatus(request.getStatus() +
               "[Transporter async.]");                             2608
     //start an inner class thread
     TransporterThread t = new TransporterThread(request, caller);
     runningThreads.put(request, t);
     t.start();
}
```

*FIG. 26G*

2610

Transporter.TransporterThread:AUS8-1999-0693

```
    /**
    * Private class to handle executions of submits() on another
thread.
    **/
    private class TransporterThread extends Thread {
        /**
        * The current request
        /**
        private RequestEvent request;

        /**
        * The caller of submit that we will call back
        **/
        private RequestResponseListener caller;


        /**
        * Create a transporter thread
        **/
        public TransporterThread(RequestEvent request,
RequestResponseListener caller) {
            super();
            this.request = request;
            this.caller = caller;
        }
        /**
        * Just call the synchronous version of
requestEventPerformed()
        **/
        public void run() {
            try {
                requestEventPerformed(request);
                caller.requestResponse(request);
            } catch (RequestException yikes); {
                caller.requestException(yikes);
            } finally {
                runningThreads.remove(request);
            }
        }
    }
```

*FIG. 26H*

Destination

FIG. 27A

2700

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 27B

2702

Methods

| Name | Declaration | Description |
|---|---|---|
| getTimeout | public abstract long getTimeout() | Invoked to return the timeout value. |
| requestEventPerformed | public abstract void requestEventPerformed (RequestEvent request) throws RequestException | Invoked to process a RequestEvent. |
| setTimeout | public abstract void setTimeout(long timeout) | Invoked to set the timeout value in ms. |

DestinationImpl

FIG. 28A

2800

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

FIG. 28B

2802

Constructors

| Name | Declaration | Description |
|---|---|---|
| DestinationImpl | public DestinationImpl() | Default constructor. |

2804

Methods

| Name | Declaration | Description |
|---|---|---|
| clear | public void clear() | By default, do nothing. |
| exit | public void exit() | By default, do nothing. |
| getJTCs | public Vector getJTCs() | By default, do nothing. |
| getTimeout | public long getTimeout() | Return the timeout value. |
| init | public void init() | By default, do nothing. |
| isEnabled | public boolean isEnabled() | Is the Destination enabled? |
| requestEventPerformed | public void requestEventPerformed (RequestEvent request) throws RequestException | A RequestEvent has arrived. If not enabled, throw an exception. Subclasses can call this method first to see if processing should continue. |
| setEnabled | public void setEnable(boolean enable) | Enable or disable the Destination. A Destination that is called when disabled will throw a RequestException. By default, record it. |
| setTimeout | public void setTimeout(long timeout) | Set the timeout value. By default, record it. |
| toString | public String toString() | Returns a String that represents the value of this object which is the class name and time timeout value. |

*FIG. 28C*

RemoteDestination.requestEventPerformed(RequestEvent):AUS8-1999-0704

```
/**
 * Process request event.
 *
 * <P>PRE: None
 * <P>POST: None
 *
 * @param request the RequestEvent object to be processed.
 * @exception RequestException if there was an error during the
 *                              processing of the event.
 */
public void requestEventPerformed(RequestEvent request) throws
RequestException {
      try {
            Method method = null;
            if (session == null) {
                  // get home interface.
                  Context ctxt = getInitialContext();
                  Object home = ctxt.lookup(request.getMajor() +
"SessionHome");

                  method = home.getClass().getMethod("create", null);
                  session = method.invoke(home, null);
            }

            //get method on home object and invoke it.
            method = session.getClass().getMethod(request.getMinor(),
                  new Class[] {Object.class});
            request.setData(method.invoke(session, new Object[]
{request.getData()}));

            if (request.getMinor().equals("remove")) {
                session = null;
            }
      } catch (InvocationTargetException te) {
            throw new RequestException(te.getTargetException());
      } catch (Throwable t) {
            throw new RequestException(t);
      }
}
```

2806

*FIG. 28D*

Factory

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |

*FIG. 29A* 2900

FIG. 29B

2902

Methods

| Name | Declaration | Description |
|---|---|---|
| list | public static void list() | Show the contents of the singletons. |
| newInstance | public static Object newInstance(String classname) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given a class name, create it and return it. |
| newInstance | public static Object newInstance(String classname, String key, boolean singleton) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given a class name, create the object and return it. If you want to create a singleton (true), then check to see if the object was already created and if so, return it. The class name is not used as the key but the 'key" parameter is. |
| newInstance | public static Object newInstance(String classname, boolean singleton) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given a class name, create the object and return it. If you want to create a singleton (true), then check to see if the object was already created and if so, return it. Use the class name as the key. |
| newInstances | public static Vector newInstances(String classnames[]) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given some class names, create and return a Vector of objects. |
| newInstances | public static Vector newInstances(String classnames[], String keys[], boolean singleton) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given some class names, create and return a Vector of objects. If you want singleton objects system wide, then if any of the classes were already created, return them, otherwise, create the new ones, remember them and return them. The class names are not used as the keys but the "keys" parameters are. |
| newInstances | public static Vector newInstances(String classnames[], boolean singleton) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given some class names, create and return a Vector of objects. If you want singleton objects system wide, then if any of the classes were already created, return them, otherwise, create the new ones, remember them and return them. Use the class name as the key. |
| removeInstance | public static void removeInstance(String key) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given a class key, clear the reference to it. |
| removeInstances | public static void removeInstances(String keys[]) throws ClassNotFoundException, InstantiationException, IllegalAccessException | Given some class keys, clear the references. |

Interface com.ibm.jtc.JTC

FIG. 30A

3000

Variables

| Name | Declaration | Description |
|---|---|---|
| _copyright | public static final String_copyright | (c) International Business Machines Inc., 1997 1998 1999. All rights reserved. |
| _version | public static final String_version | |
| _author | public static final String_author | |
| _email | public static final String_email | |

FIG. 30B

3002

Methods

| Name | Declaration | Description |
|---|---|---|
| clear | public abstract void clear() | Invoked to indicate that all memory allocations should be cleaned up. This includes removing listeners and flushing any lists (vectors or hashtables). A JTC object that has been cleared can be reused. |
| exit | public abstract void exit() | Invoked to indicate that all memory allocations should be cleaned up. This includes removing listeners and flushing any lists (vectors or hashtables). It also includes setting all variable references to null. A JTC object that has been cleared cannot be reused. |
| getJTCs | public abstract Vector getJTCs() | Invoked to get a Vector of all JTC objects that this JTC object has created. For example, a Transporter will at least return all of its Destinations. This is a very powerful mechanism. It allows us to get a reference to all primary objects in the JTC application and manipulate them according to the JTC methods, or by casting them to more specific classes or interfaces and manipulating them. Examples usage includes non code intrusive tracing, debugging, logging, profiling, etc. |
| init | public abstract void init() | Invoked to initialize the JTC object. The object should be ready for operation. |
| isEnabled | public abstract boolean isEnabled() | Invoked to determine if the JTC object is enabled. |
| setEnabled | public abstract void setEnabled (boolean enable) | Invoked to enable or disable the JTC object. |
| toString | public abstract String toString() | Invoked to get a String representation of the JTC object. |

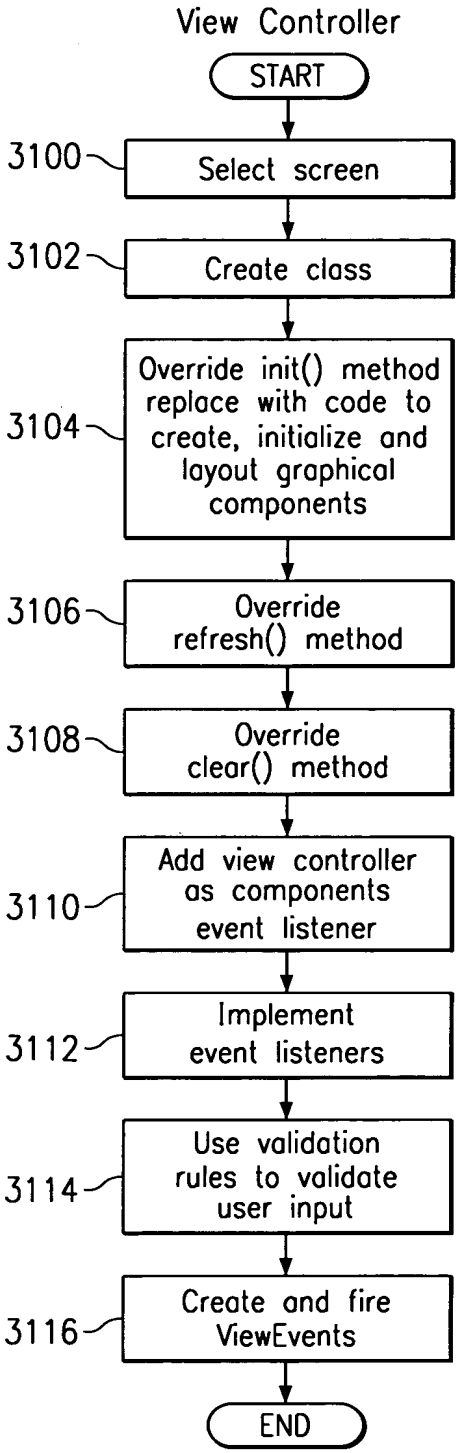



FIG. 31

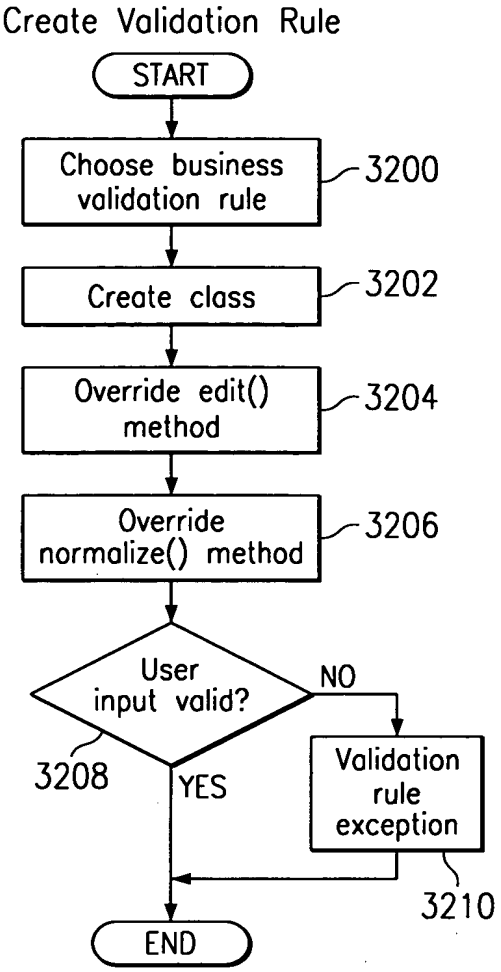



FIG. 32

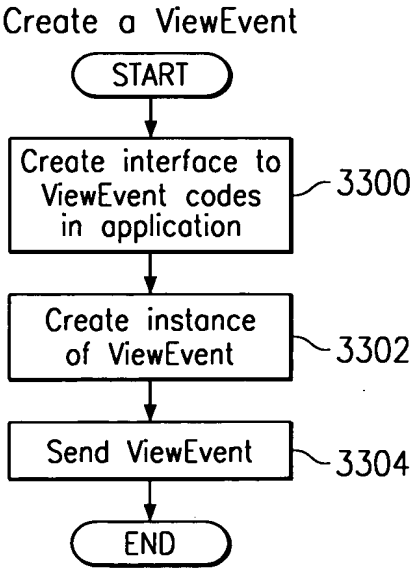



FIG. 33

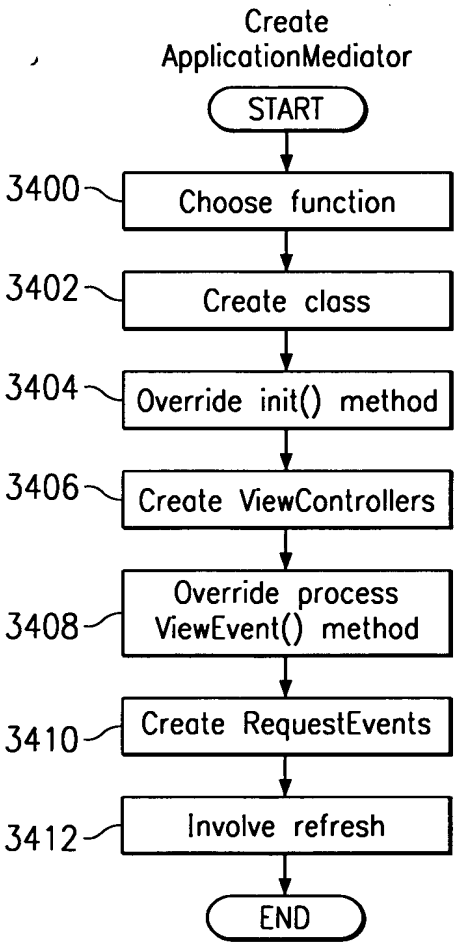



FIG. 34

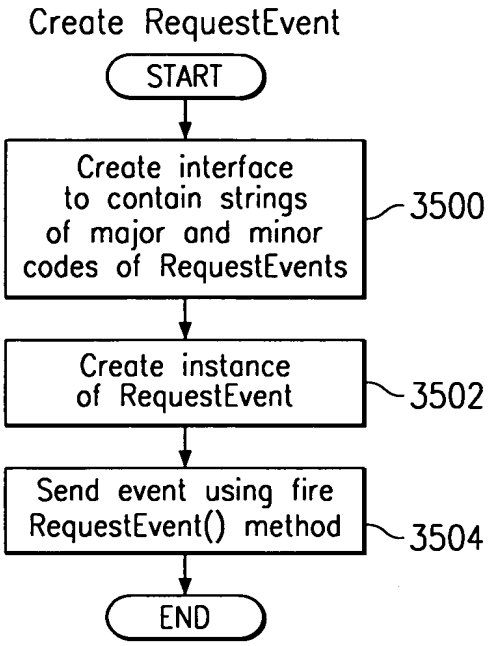



FIG. 35

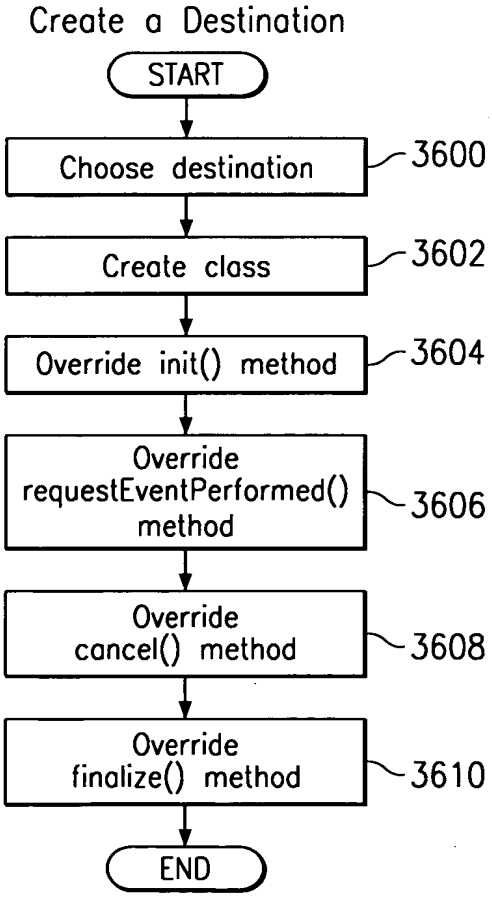



FIG. 36

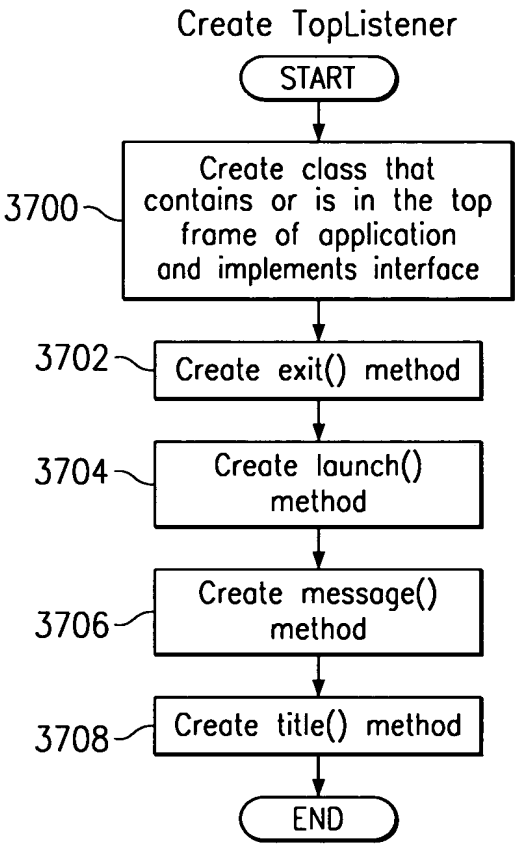

Create TopListener
START
Create class that contains or is in the top frame of application and implements interface
3700
Create exit() method
3702
Create launch() method
3704
Create message() method
3706
Create title() method
3708
END

*FIG. 37*

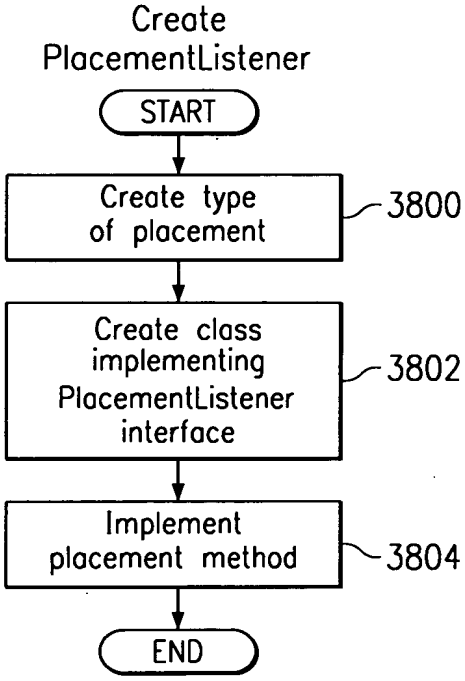

Create PlacementListener
START
Create type of placement
3800
Create class implementing PlacementListener interface
3802
Implement placement method
3804
END

*FIG. 38*

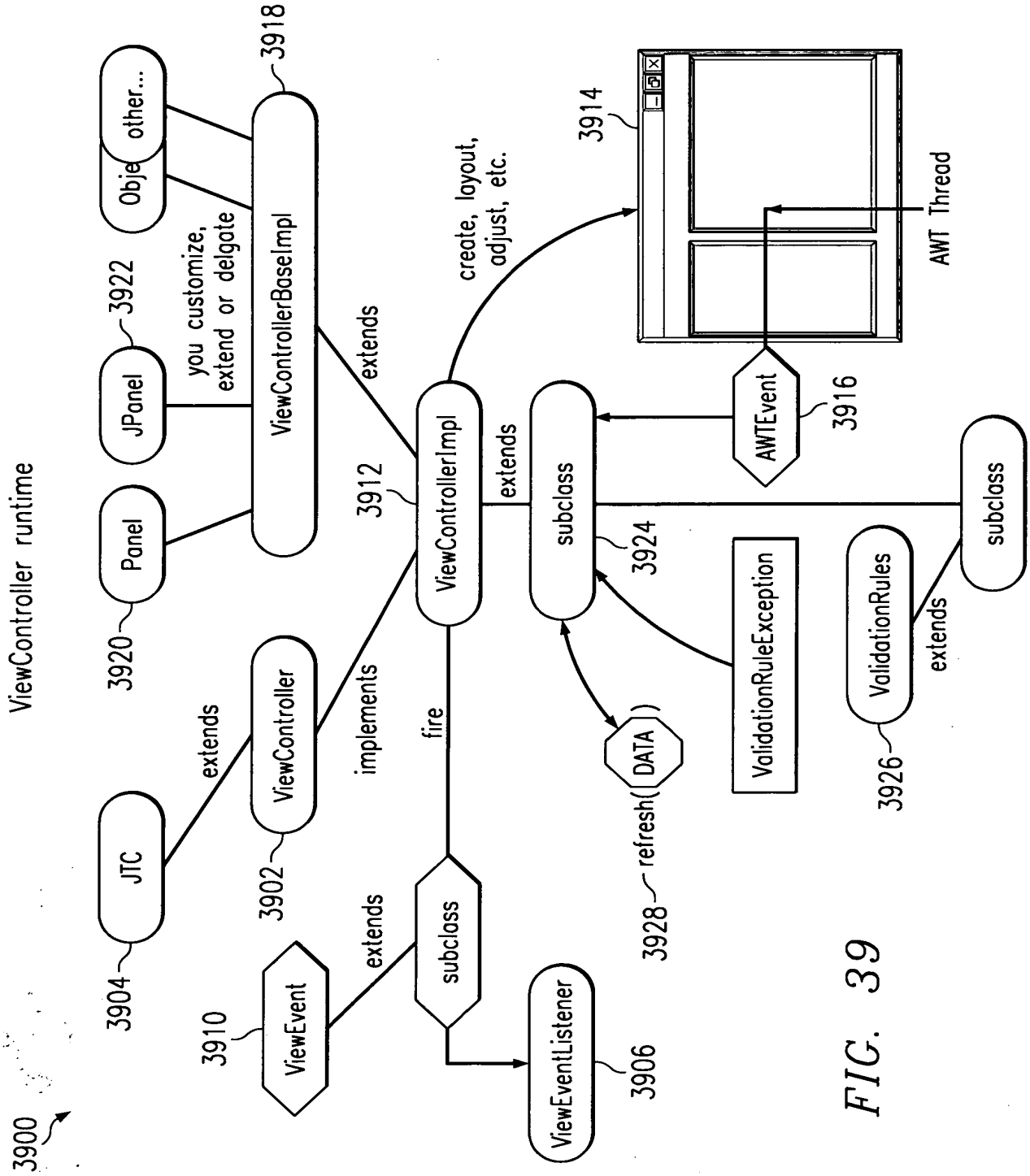

ViewController runtime
3900
JTC
3904
ViewEvent
3910
ViewController
3902
Panel
3920
JPanel
3922
Obje
other...
ViewControllerBaseImpl
3918
you customize,
extend or delgate
extends
implements
extends
fire
subclass
ViewEventListener
3906
refresh(
DATA
3928
ViewControllerImpl
3912
extends
create, layout,
adjust, etc.
3914
AWT Thread
AWTEvent
3916
subclass
3924
ValidationRuleException
ValidationRules
3926
extends
subclass

FIG. 39

Basic Operation of a ViewControllerImpl

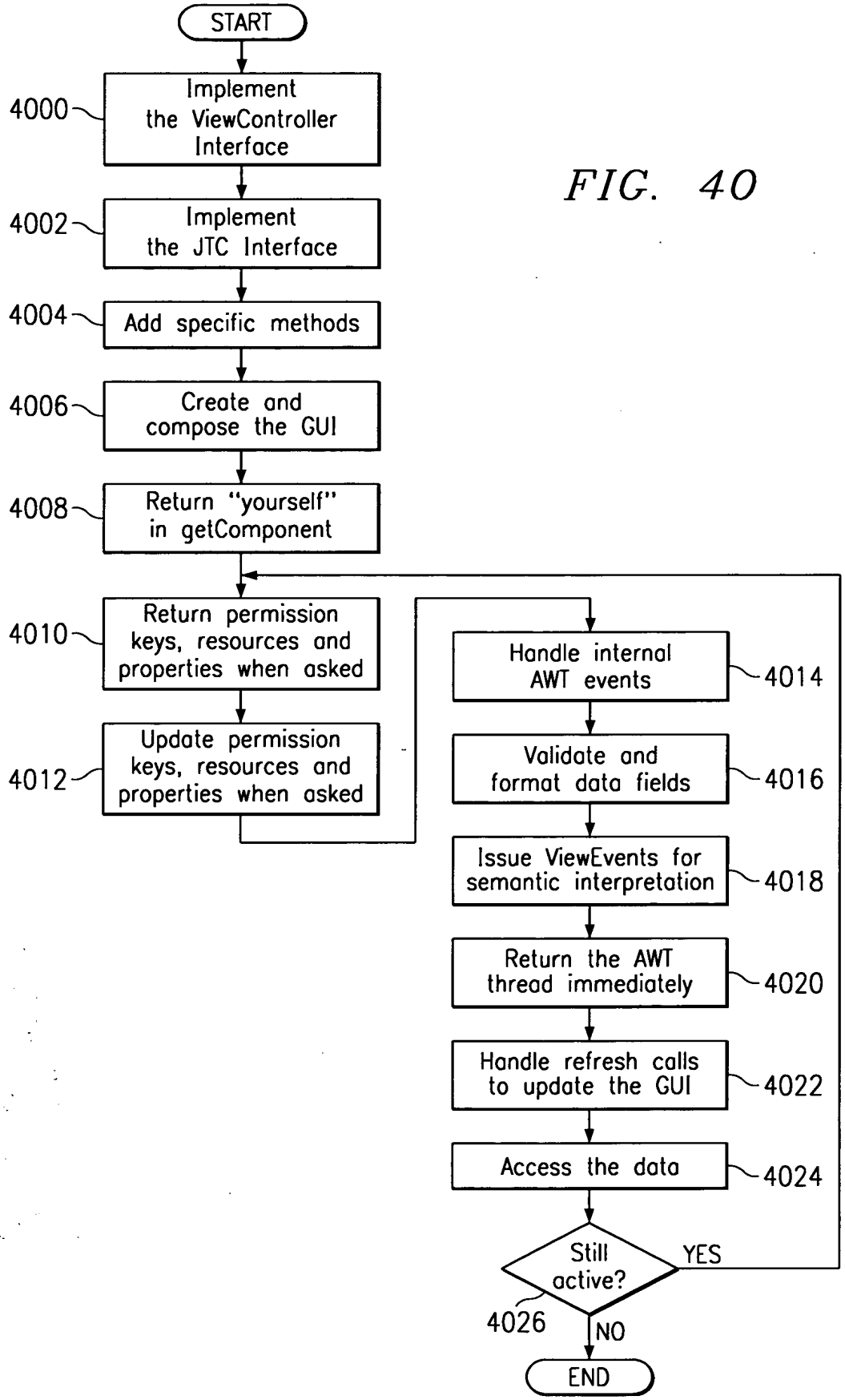



*FIG. 40*

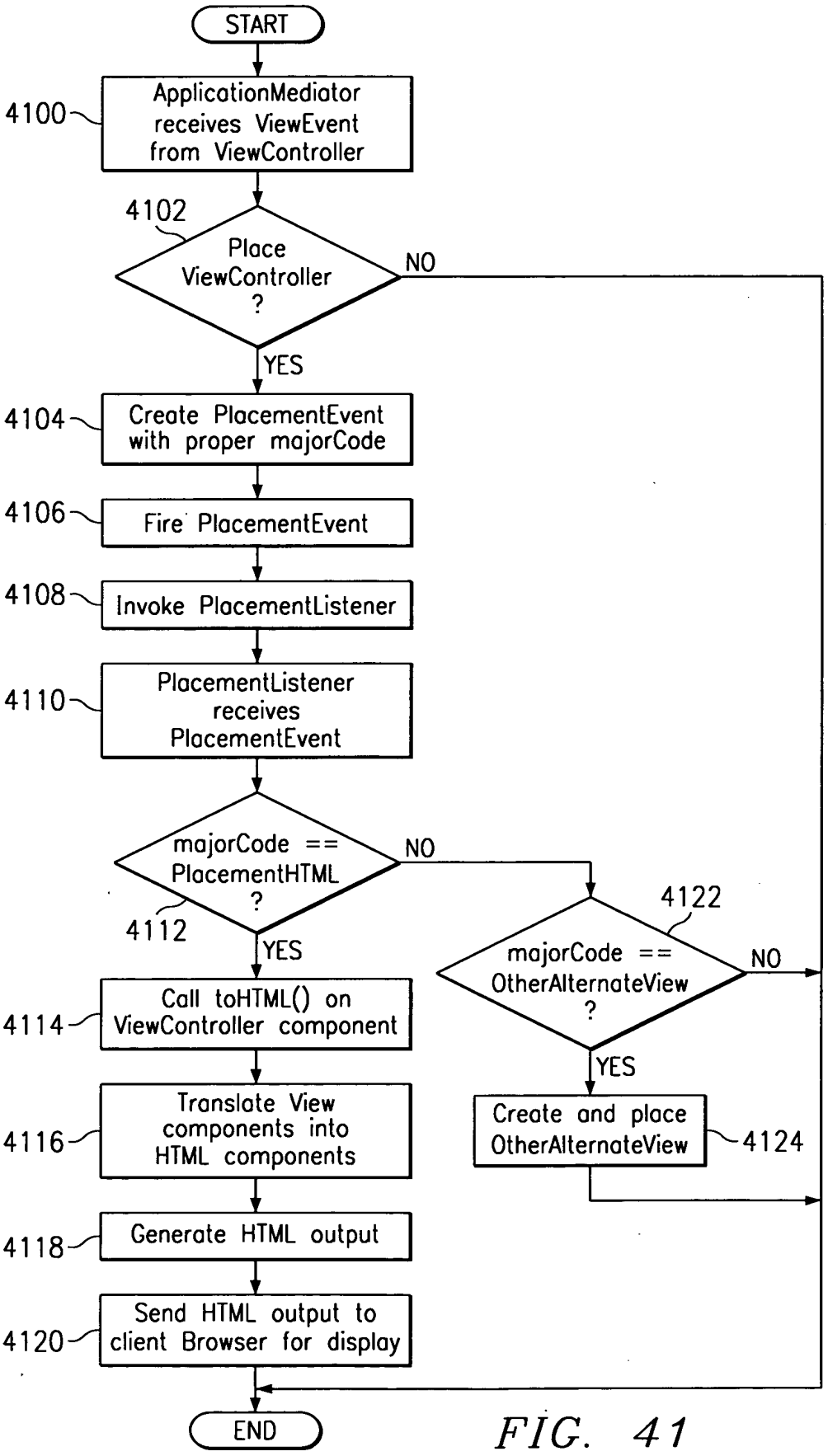

START
4100
ApplicationMediator receives ViewEvent from ViewController
4102
Place ViewController ?
NO
YES
4104
Create PlacementEvent with proper majorCode
4106
Fire PlacementEvent
4108
Invoke PlacementListener
4110
PlacementListener receives PlacementEvent
majorCode == PlacementHTML ?
NO
4112
YES
4114
Call toHTML() on ViewController component
4116
Translate View components into HTML components
4118
Generate HTML output
4120
Send HTML output to client Browser for display
4122
majorCode == OtherAlternateView ?
NO
YES
Create and place OtherAlternateView
4124
END

FIG. 41

ViewEvent and ViewListener Usage

```
→ Usage from a ViewController
  public void actionPerformed(ActionEvent e) {
      if (e.getSource() == nextButton) {
          ViewEvent ve = new ViewEvent (this);
          ve.setMajor(ViewEvent.NEXT);
          fireViewEvent(ve); //notify
      ViewEvent listener
          return;
      }
  }
```

*FIG. 42*

```
→ Usage from ViewListener (i.e. ApplicationMediator)
  //add myself as a listener
  customerDetailsViewController.addViewListener(this);

  //later, we are called back on this method to handle the
ViewEvent
  processViewEvent (ViewEvent event) {
      //do something
      switch (event.getMajor()) {
         case ViewEvent.NEXT: //...
         break;
         case ViewEvent.OK: //...
         break;
      }
  }
```

*FIG. 43*

Major and/or minor codes

→ Pre-defined major codes- A subclass can define others.

- // system
  - OK DONE OPEN CLOSE CANCEL EXIT FILE SAVE SAVEAS ERROR WARNING RETURN LOAD NOTIFY NOTIFY2 INFO SETUP PRINT LOGIN LOGOUT ENABLE DISABLE
- // status
  - TITLEMESSAGE STATUSMESSAGE ERRORMESSAGE SUGGESTIONMESSAGE
- // navigational
  - NEXT PREVIOUS FIRST LAST START BEGIN END PAUSE STOP RESTART SUBMIT BACKSPACE INSERT HOME PGUP PGDN LEFT RIGHT UP DOWN
- // live
  - FAST MEDIUM SLOW RUN DELAY WAIT TIMER ON OFF HIGH LOW
- // data related
  - LIST MORE ADD DELETE MODIFY NEW EDIT COPY CUT PASTE UNDO REMOVE PLUS MINUS INCREMENT DECREMENT CHANGED FILL EMPTY READY VIEW DETAILS READ WRITE UPDATE REFRESH
- // assit related
  - SEARCH FIND HELP HINT TRAIN TEACH SUGGEST
- // sub options related
  - A B C D E F OPTION CHOOSE
- // test values
  - TRACE UNTRACE DEBUG UNDEBUG LOG UNLOG HOOK UNHOOK
- // ibm values

TEAM WIN EXECUTE

*FIG. 44*

ValidationRules Usage

→ Examples:

```
edit("123456") -> $1234.56
normalize("$1234.56") -> 123456
edit("12345x") -> ValidationRuleException
```

→ edit

```
//validate and re-display
String value = textfield.getText();
try {
    result = SocialSecurity.edit(value);
}
catch (ValidationRuleException yikes) {
    //...
    return;
}
textField.setText(value);
```

*FIG. 45*

→ normalize

```
//validate and update the data objects
String value = textfield.getText();
try {
    result = SocialSecurity.normalize(value);
}
catch (ValidationRuleException yikes) {
    //message box ...
    return;
}
dataObject.setText(value);
```

*FIG. 46*

ValidationRules Usage

→ Example Chaining

```
//each rule
String range = "com.xyz.jtc.RangeChecker";
String money = "com.xyz.jtc.AccountMoney";

//build the chain of rules
String[] rules = {range, money};

//get the value to validate
String value = textField.getText();

try {
    value = applyEdits(rules, input);

}
catch (ValidationRuleException ouch) {
    //...
}

//the value is validated and formatted, redisplay
textField.setText(value);
```

*FIG. 47*

ViewControllerBaseImpl

→ For example:

• inheritance

```
public class ViewControllerBaseImpl extends JPanel {
    public Component getComponent() {
    return this;
    }
}
```

*FIG. 48*

• delegation

```
public class ViewControllerBaseImpl implements ViewController
{
    XYZ xyz = new XYZ();

    public java.awt.Component getComponent() {
        return xyz;
    }

    public void setEnabled(boolean e) {
        xyz.setEnabled(e);
    }

    public void setVisible(boolean v) {
        xyz.setVisible(v);
    }
}
```

*FIG. 49*

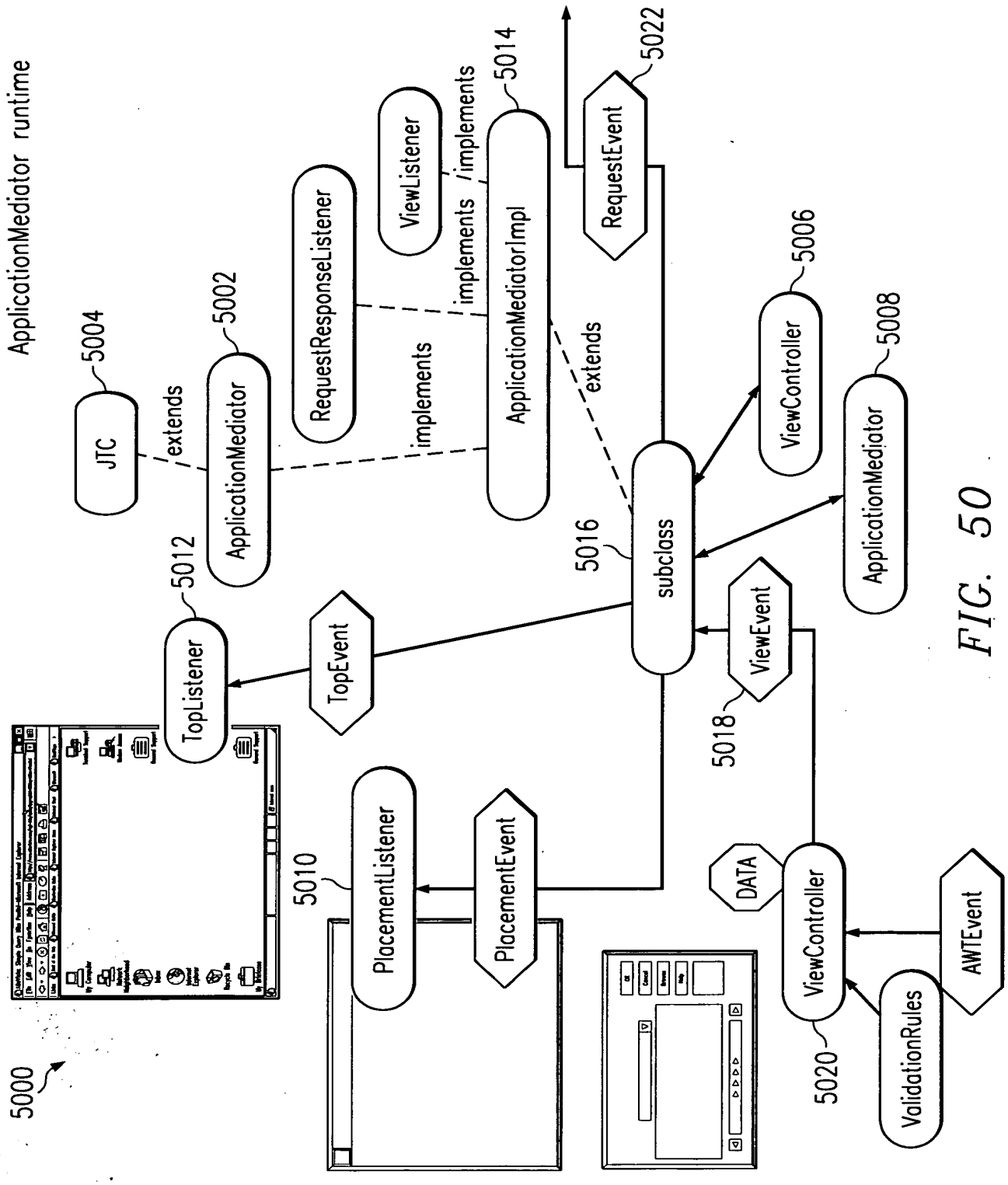

ApplicationMediator runtime
5000
5004
JTC
extends
5002
ApplicationMediator
RequestResponseListener
implements
ViewListener
implements
implements
5014
ApplicationMediatorImpl
extends
5022
RequestEvent
5006
ViewController
5008
ApplicationMediator
5012
TopListener
TopEvent
5016
subclass
ViewEvent
5018
5010
PlacementListener
PlacementEvent
DATA
ViewController
5020
ValidationRules
AWTEvent

FIG. 50

AWTEvent threading support

- Style 1 – wait/Queue/notify
- Style 2 – Thread dispatch
- Handles Threading Model for ViewControllers

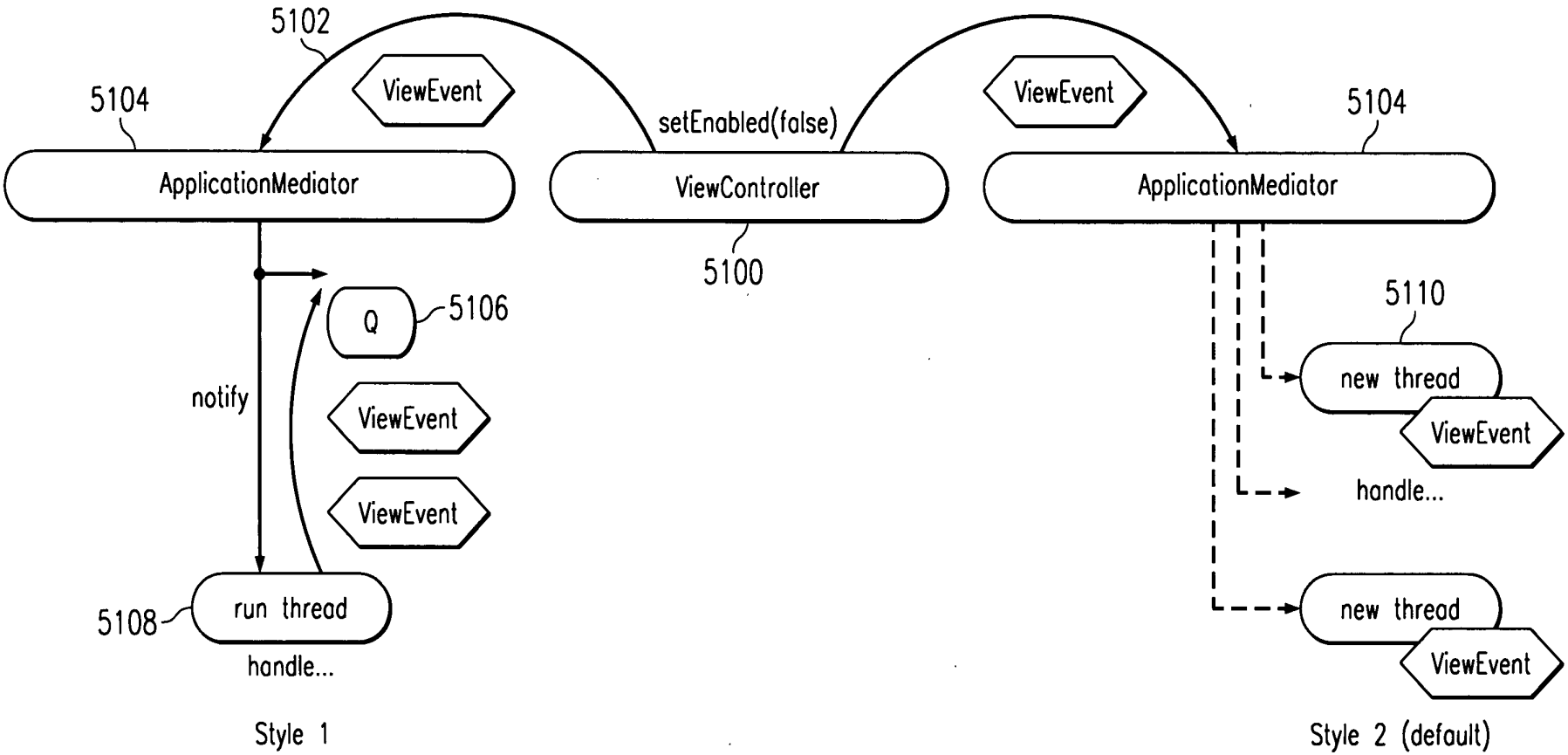



*FIG. 51*

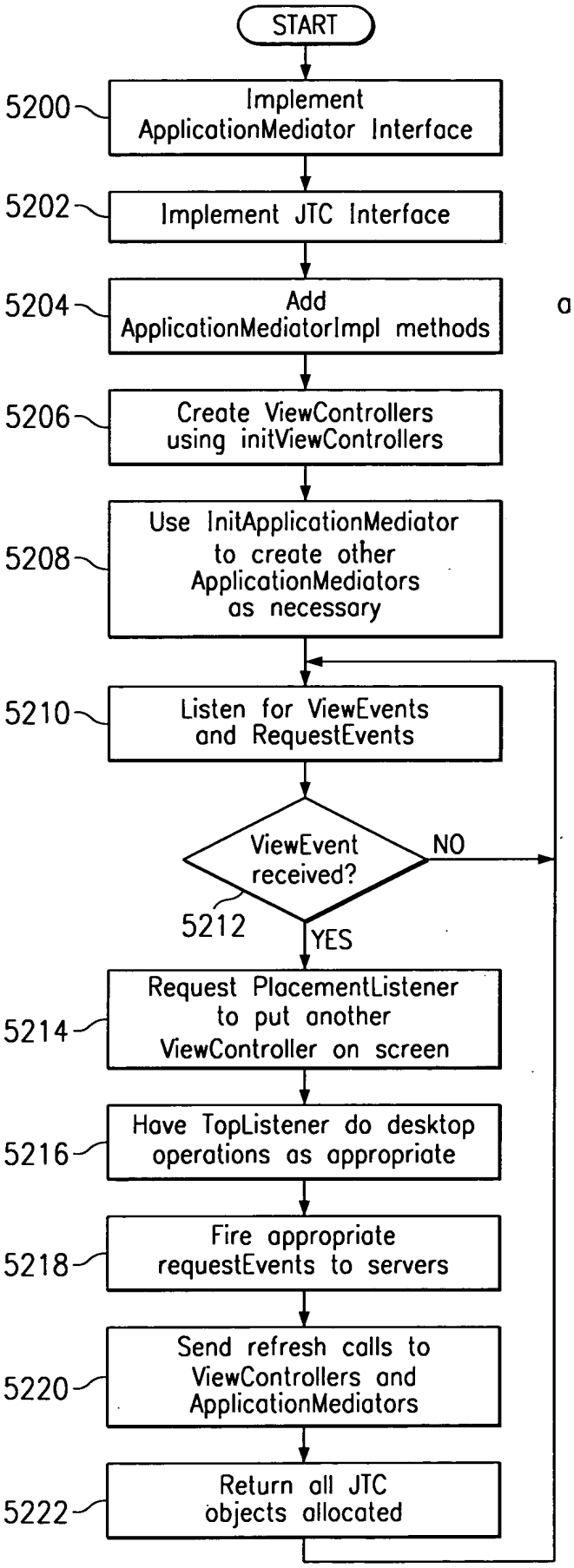



*FIG. 52*

Design Steps for an ApplicationMediator

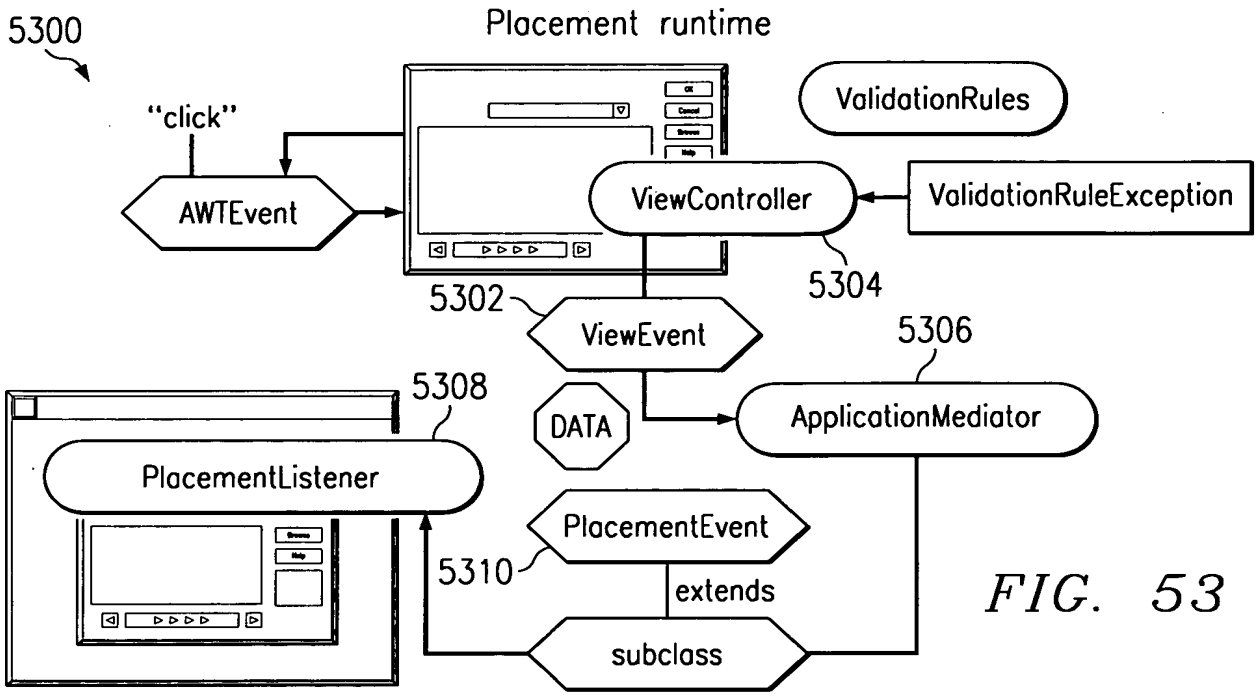



FIG. 53

Placement example

```
Usage from ApplicationMediator
        //in an ApplicationMediator
        int major = PlacementEvent.ADD;
        Component component =
        customerDetailsViewController.getComponent();
        PlacementEvent e = new PlacementEvent(this, component, major);
        firePlacementEvent(e);
```

FIG. 54

```
Usage from PlacementListener
        public class MyProgram implements PlacementListener {
            public void placementEventPerformed(PlacementEvent e) {
                //decide based on source type
                switch (e.getMajor()) {

                    case PlacementEvent.ADD:
                        if (e.getSource() instanceof PreferencesAm)
                            panel1.add("Center", e.getComponent());
                        else panel2.add("A", e.getComponent());
                    break;
                    case PlacementEvent.REMOVE:
                        //do something else
                    break;
                }
            //etc.
            }
        }
```

FIG. 55

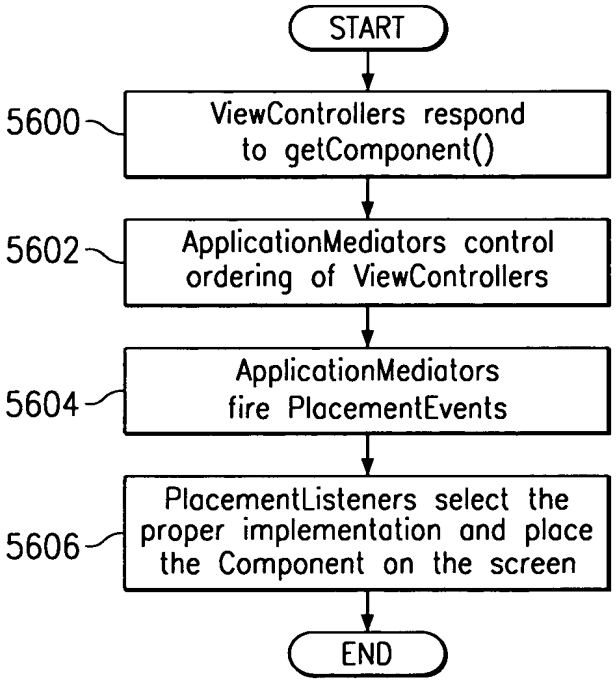



*FIG. 56*

Design Steps for a PlacementEvent

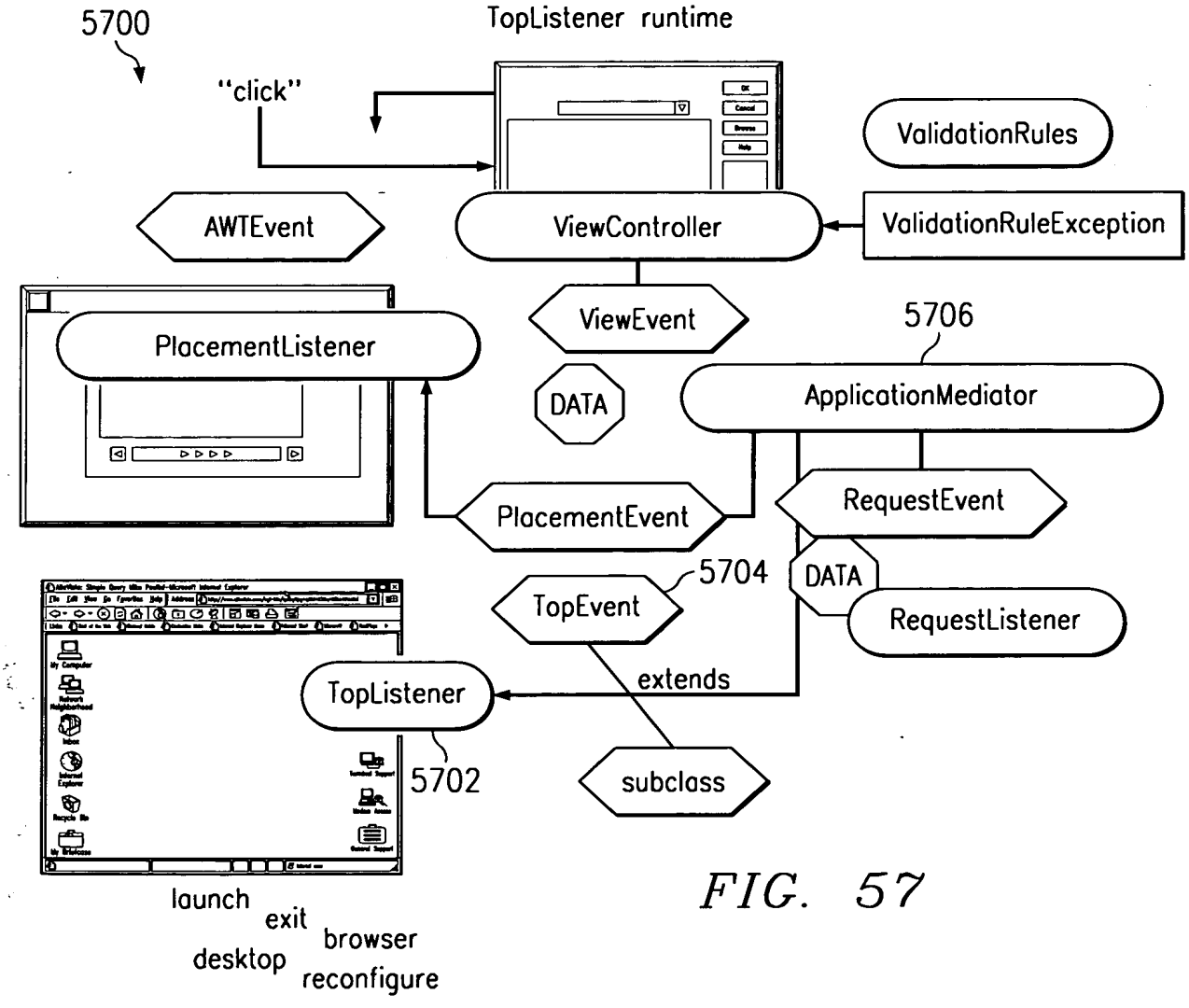



*FIG. 57*

TopListener example

```
//from the TopListener
ApplicationMediatorXYZ m = new ApplicationMediatorXYZ();
m.addTopListener(this);
```

*FIG. 58*

```
//in the ApplicationMediator
String status = "Loading files...";
TopEvent e = new TopEvent(this, TopEvent.STATUS, 0, status);
fireTopEvent(e);
```

*FIG. 59*

```
//later in the TopListener callback
public void topEventPerformed(TopEvent e) {
    switch(e.getMajor()) {
        case STATUS:
            //access the browser
        break;
        /etc.
    }
}
```

*FIG. 60*

RequestEvent example

```
//from an ApplicationMediator - create event
RequestEvent r = new RequestEvent();
r.setMajor ("Loans");
r.setMinor("SubmitCustomerInfo");
```

*FIG. 62*

```
//fire an asynchronous event
try {
    //asynchronous
    fireRequestEvent(this, r);
}
catch (RequestException yikes) {}
```

*FIG. 63*

```
//later, called back with success
public void requestResponse(RequestEvent result) {
    //process response
}

//or failure
public void requestException(RequestException yikes) {
    //now what?
}
```

*FIG. 64*

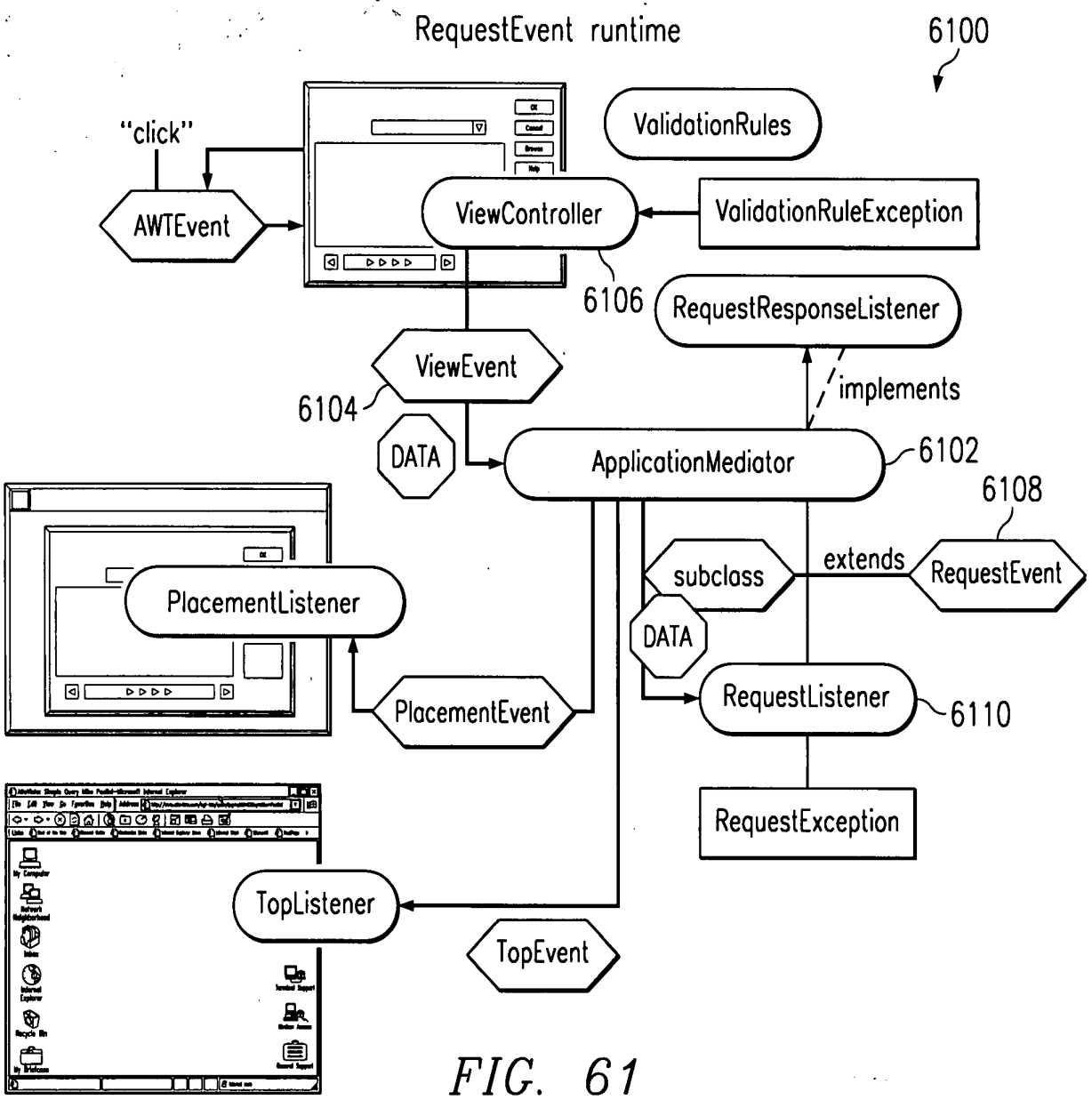

RequestEvent runtime
6100
"click"
AWTEvent
ViewController
ValidationRules
ValidationRuleException
6106
RequestResponseListener
ViewEvent
6104
implements
DATA
ApplicationMediator
6102
6108
PlacementListener
subclass
extends
RequestEvent
DATA
PlacementEvent
RequestListener
6110
RequestException
TopListener
TopEvent

FIG. 61

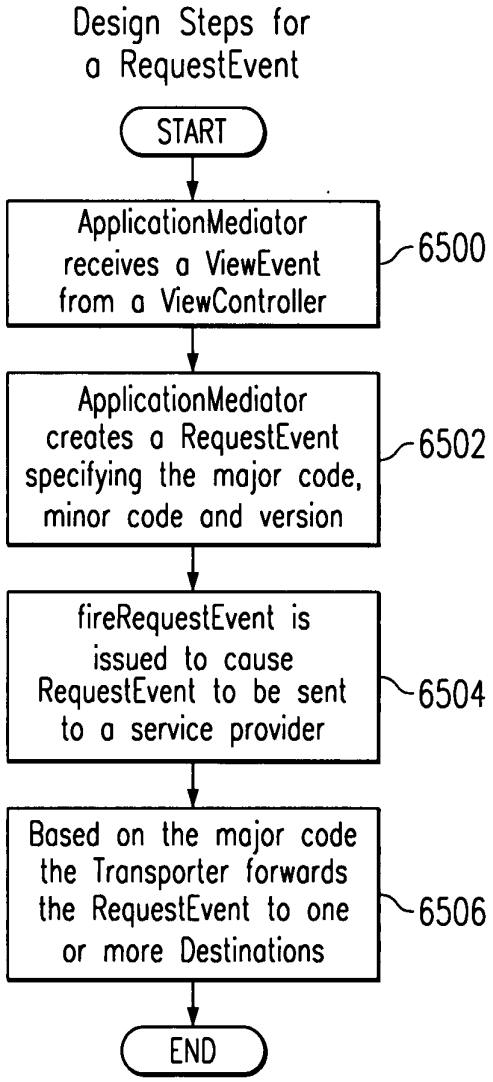

Design Steps for a RequestEvent
START
ApplicationMediator receives a ViewEvent from a ViewController
6500
ApplicationMediator creates a RequestEvent specifying the major code, minor code and version
6502
fireRequestEvent is issued to cause RequestEvent to be sent to a service provider
6504
Based on the major code the Transporter forwards the RequestEvent to one or more Destinations
6506
END

FIG. 65

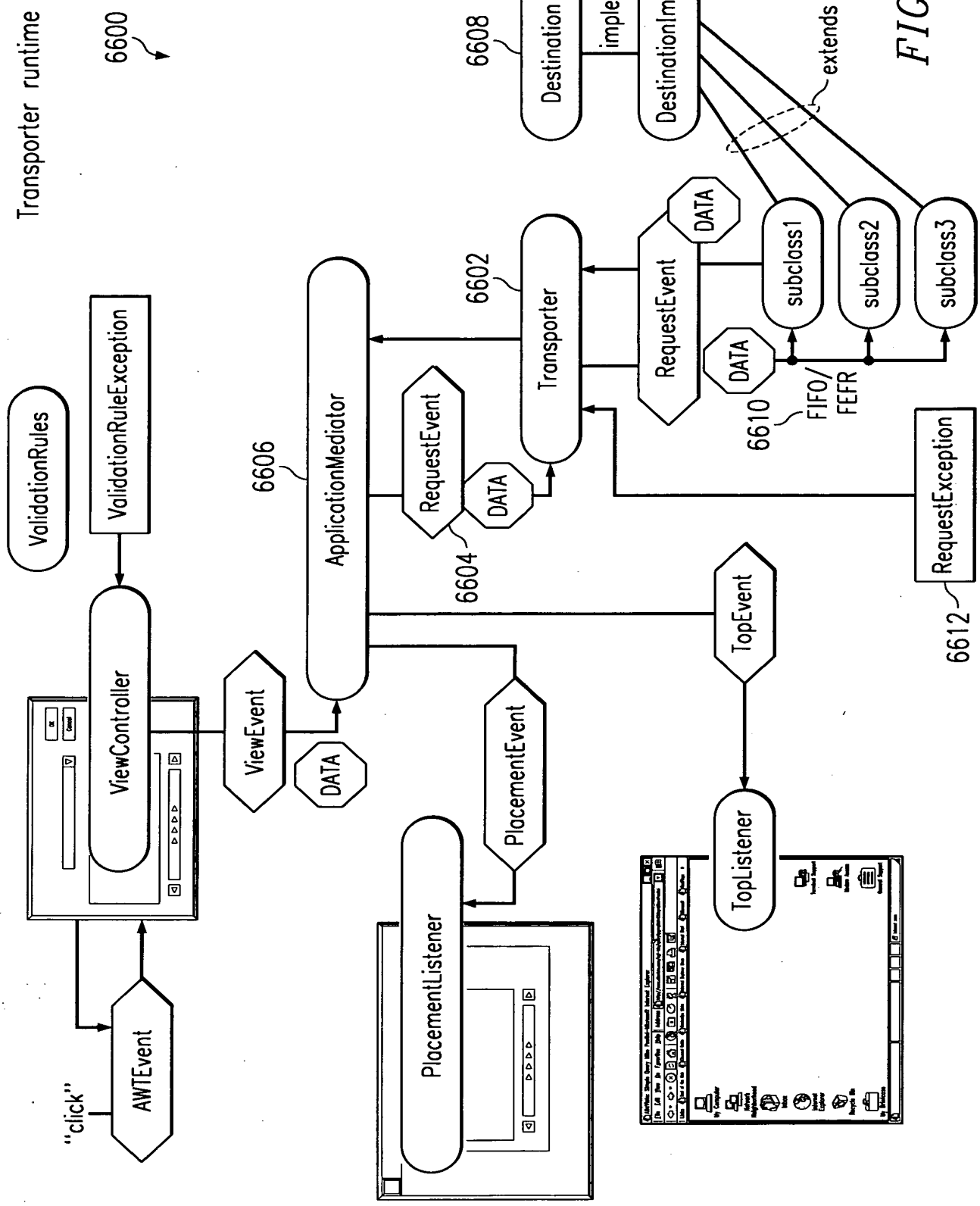

Transporter runtime
6600
ValidationRules
ValidationRuleException
ViewController
"click"
AWTEvent
ViewEvent
DATA
6606
ApplicationMediator
PlacementEvent
PlacementListener
6604
RequestEvent
DATA
6602
Transporter
RequestEvent
DATA
DATA
6610
FIFO/
FEFR
subclass1
subclass2
subclass3
6608
Destination
implements
DestinationImpl
extends
RequestException
6612
TopEvent
TopListener

FIG. 66

Transporter

- This class implements the JTC and RequestEventListener interfaces
- Its primary function is to map RequestEvents to Destinations.
  - Typically ApplicationMediators fire RequestEvents and Destinations process them
- Add a Transporter to an ApplicationMediator to listen for RequestEvents

```
Transporter t = new Transporter();
ApplicationMediator am = new ApplicationMediator();
am.addRequestListener(t);
```

- The ApplicationMediator will fire RequestEvents

```
RequestEvent r = new RequestEvent(source, major, minor, data);
try {
    fireRequestEvent(r);
}
catch (RequestException yikes) {}
```

*FIG. 67*

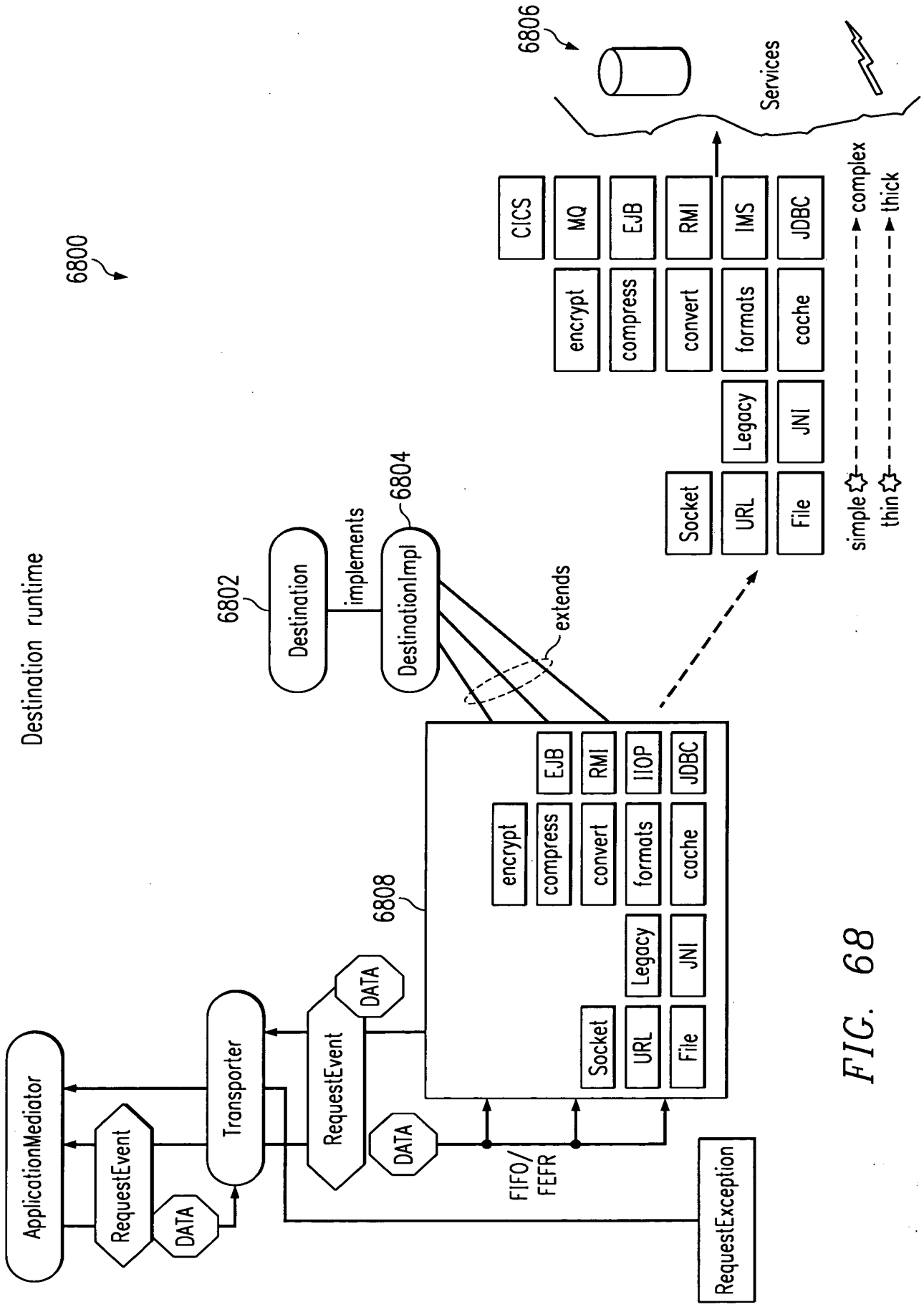

6800
Destination runtime
6802
Destination
implements
6804
DestinationImpl
extends
6806
Services
6808
ApplicationMediator
RequestEvent
DATA
Transporter
RequestEvent
DATA
DATA
FIFO/
FEFR
RequestException
Socket
URL
File
Legacy
JNI
encrypt
compress
convert
formats
cache
EJB
RMI
IIOP
JDBC
CICS
MQ
EJB
RMI
IMS
JDBC
encrypt
compress
convert
formats
cache
Socket
URL
File
Legacy
JNI
simple
thin
complex
thick

FIG. 68

## Destination

- RequestEvents are identified by
  - major code – represents a family of Requests
  - minor code – represents a specific Request
- Destinations are added to the Transporter as DestinationListeners specifing a major code for RequestEvents they are interested in receiving
- The destination is called when the major code of the RequestEvent matches the destination major code

```
EJBDestination d = new EJBDestination();
Transporter t = new Transporter();
String major = "Loans";
t.addDestinationListener(major, d);
```

- Multiple Destinations can listen for the same RequestEvent major code
  - processed FIFO/FESP (first in first out, first exception stop forwarding)
  - results of one Destination can be passed to the next Destination

*FIG. 69*

## Destinations and major codes

- Special major codes
  - wildcard
    - "*" major code indicates the Destination is interested in all and any RequestEvents
    - processed after specific major codes have been matched.
  - priority
    - "!" major code indicates the Destination is interested in all requests and should be given priority.
    - processing performed before specific major codes and wildcards
- For example

```
Transporter t = new Transporter();
t.addDestinationListener ("*", new WildDestination ());
t.addDestinationListener ("Loans", new EJBDestination());
t.addDestinationListener ("!", new PriorityDestination());

//later
RequestEvent r = new RequestEvent(this, "Loans", " ", null);
try {
    fireRequestEvent(r);
}
catch (RequestException yikes) {}
```

- The RequestEvent "r" will be sent to PriorityDestination 1st, EJBDestination 2nd, and WildDestination() 3rd, assuming no RequestExceptions are thrown.

*FIG. 70*

getJTCs example

```
// Recursively look at the root, find each JTC and/or AWT and hook
public void hookJTCs(JTC root) {
    Vector jtcs = null;
    try {
        jtcs = root.getJTCs();
    } catch (Exception none) { return; } // should not happen

    if (jtc == null) return; //we are done

    int size = jtcs.size();
    for (int j = 0; j < size; j++) {
        Object current = jtcs.elementAt(j);
        if (current instanceof ApplicationMediator) {
            hookAM((ApplicationMediator) current);
        } else
            if (current instanceof ViewController) {
                hookVC((ViewController) current);
            } else
                if (current instanceof Transporter) {
                    hookTransporter((Transporter) current);
                } else
                    if (current instanceof java.awt.Component) {
                        //once into AWT tree, never back to JTCs
                        hookAWTs((java.awt.Component) current);
                        continue;
                    }
        hookJTCs((JTC) jtcs.elementAt(j)); //recursive
    }
}
```

FIG. 71

hookJTC helpers

```
/**
 * Hook the ApplicationMediator
 */
public void hookAM(ApplicationMediator am) {
    vc1.refresh("ApplicationControllers found:" + am);
    am.addViewListener(this);
    am.addRequestListener(this);
    vc1.refresh("....add as ViewListener");
    vc1.refresh("....add as RequestListener");
}

/**
 * Hook the ViewController and it's getComponent()
 */
public void hookVC(ViewController vc) {
    vc1.refresh("ViewController found:" + vc);
    vc.addViewListener(this);
    vc1.refresh("....add as ViewListener");
    hookAWTs(vc.getComponent());
}

/**
 * Hook the Transporter
 */
public void hookTransporter(Transporter transporter) {
    vc1.refresh("Transporter found:" + transporter);
    transporter.addDestinationListener("!", this);
    vc1.refresh("....add as ! DestinationListener");
}
```

FIG. 72

hookAWTs

```
//Recursively find each AWT object and hook
public void hookAWTs(Component comp) {
    if (component instanceof Container) {
        vc1.refresh("Container found:" + comp);
        Component[] comps = ((Container) comp).getComponents();
        int size = comps.length;
        for (int i = 0; i < size; i++) {
            hookAWTs(comps[i]);
        }
    }
    /* continue here since some regular Components, such as JLabels,
    * are Containers also.
    */
    if (comp instanceof Button) {
        hookAWTButton((Button) comp);
    } else
        if (comp instanceof JButton) {
            hookSwingJButton((JButton) comp);
        } else
            if (component instanceof JTextField) {
                hookSwingJTextField((JTextField) comp);
            }
            /*...else do over every other Bean/Component/Container
            * type possibly using reflection or a table driven
            * implementation.
            */
}
```

FIG. 73

hookAWTs – helpers

```
/**
 * Hook the java.awt.Button
 */
public void hookAWTButton(Button button) {
    vc1.refresh("java.awt.Button found:" + button);
    button.addActionListener(this);
    vcl.refresh("....add as ActionListener");
}
/**
 * Hook the com.sun.java.swing.JButton
 */
public void hookSwingJButton(JButton button) {
    vc1.refresh("com.sun.java.swing.JButton found:"+ button);
    button.addActionListener(this);
    button.addChangeListener(this);
    button.addItemListener(this);
    vc1.refresh("....add as ActionListener");
    vc1.refresh("....add as ChangeListener");
    vc1.refresh("....add as ItemListener");
}
/**
 * Hook the com.sun.java.swing.JTextField
 */
public void hookSwingJTextField(JTextField textfield) {
    vc1.refresh("com.sun.java.swing.JTextField found:" textfield);
    textfield.addActionListener(this);
    textfield.addCaretListener(this);
    vc1.refresh("....add as ActionListener");
    vc1.refresh("....add as CaretListener");
}
```

FIG. 74

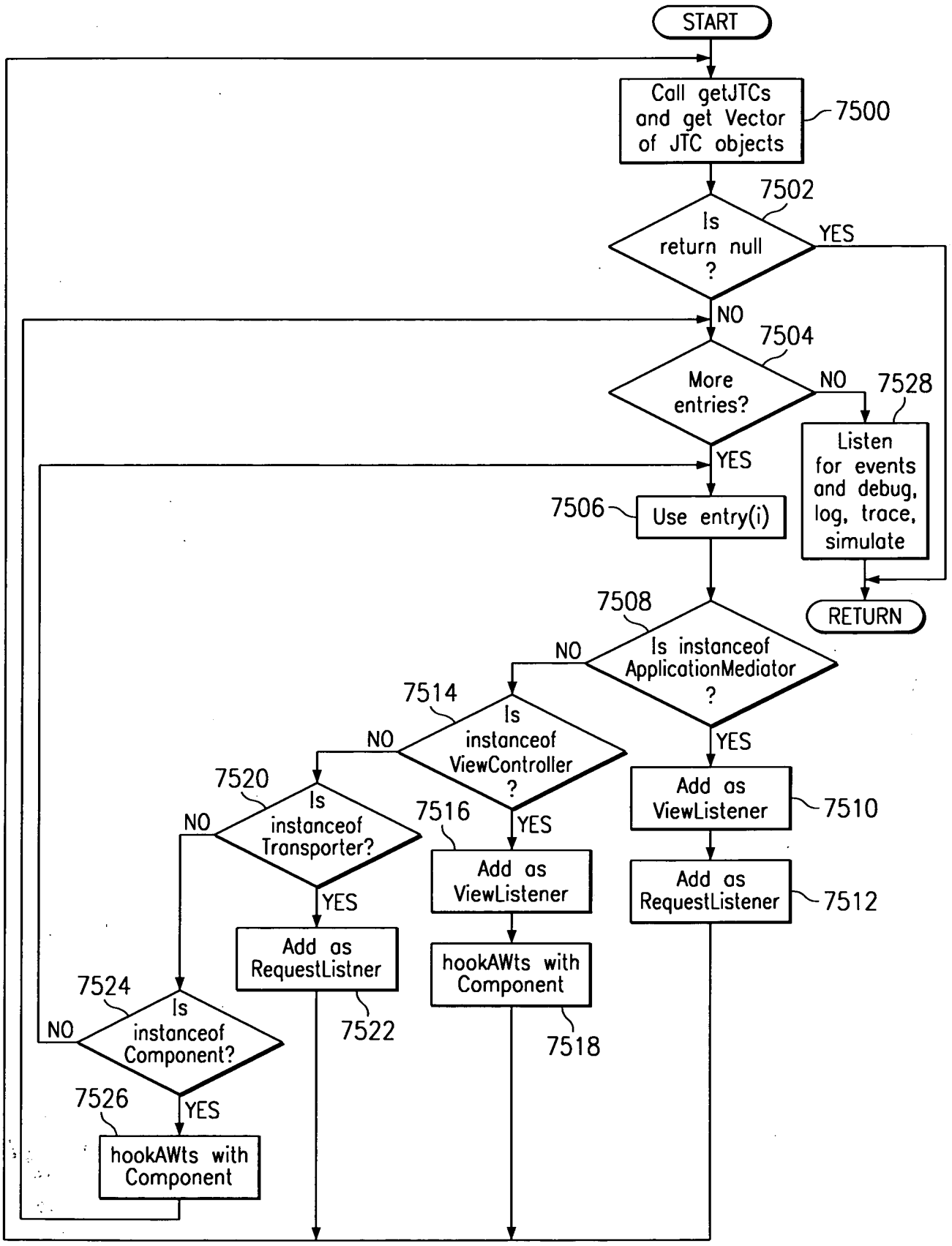

START
Call getJTCs and get Vector of JTC objects
7500
7502
Is return null ?
YES
NO
7504
More entries?
NO
7528
Listen for events and debug, log, trace, simulate
YES
7506
Use entry(i)
RETURN
7508
Is instanceof ApplicationMediator ?
NO
YES
7514
Is instanceof ViewController ?
NO
YES
Add as ViewListener
7510
7520
Is instanceof Transporter?
NO
YES
7516
Add as ViewListener
Add as RequestListener
7512
Add as RequestListner
hookAWts with Component
7522
7518
7524
Is instanceof Component?
NO
YES
7526
hookAWts with Component

*FIG. 75*

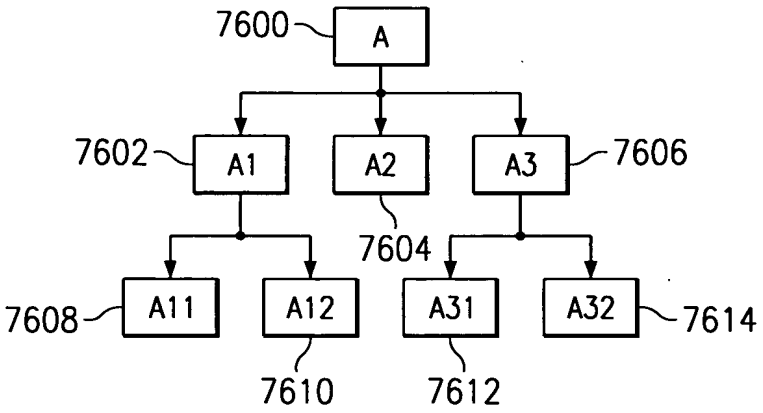

7600
A
7602
A1
A2
A3
7606
7604
7608
A11
A12
A31
A32
7614
7610
7612

*FIG. 76*

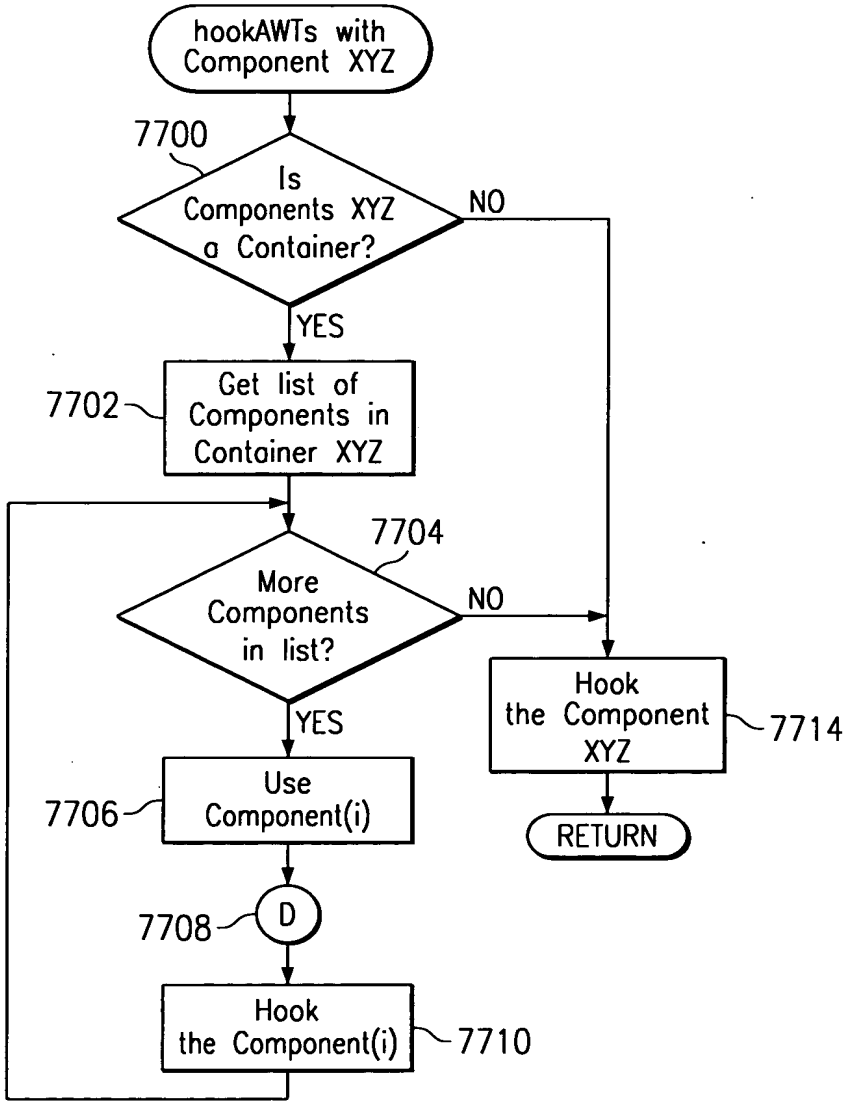

hookAWTs with Component XYZ
7700
Is Components XYZ a Container?
NO
YES
7702
Get list of Components in Container XYZ
7704
More Components in list?
NO
YES
Hook the Component XYZ
7714
RETURN
7706
Use Component(i)
7708
D
Hook the Component(i)
7710

*FIG. 77*

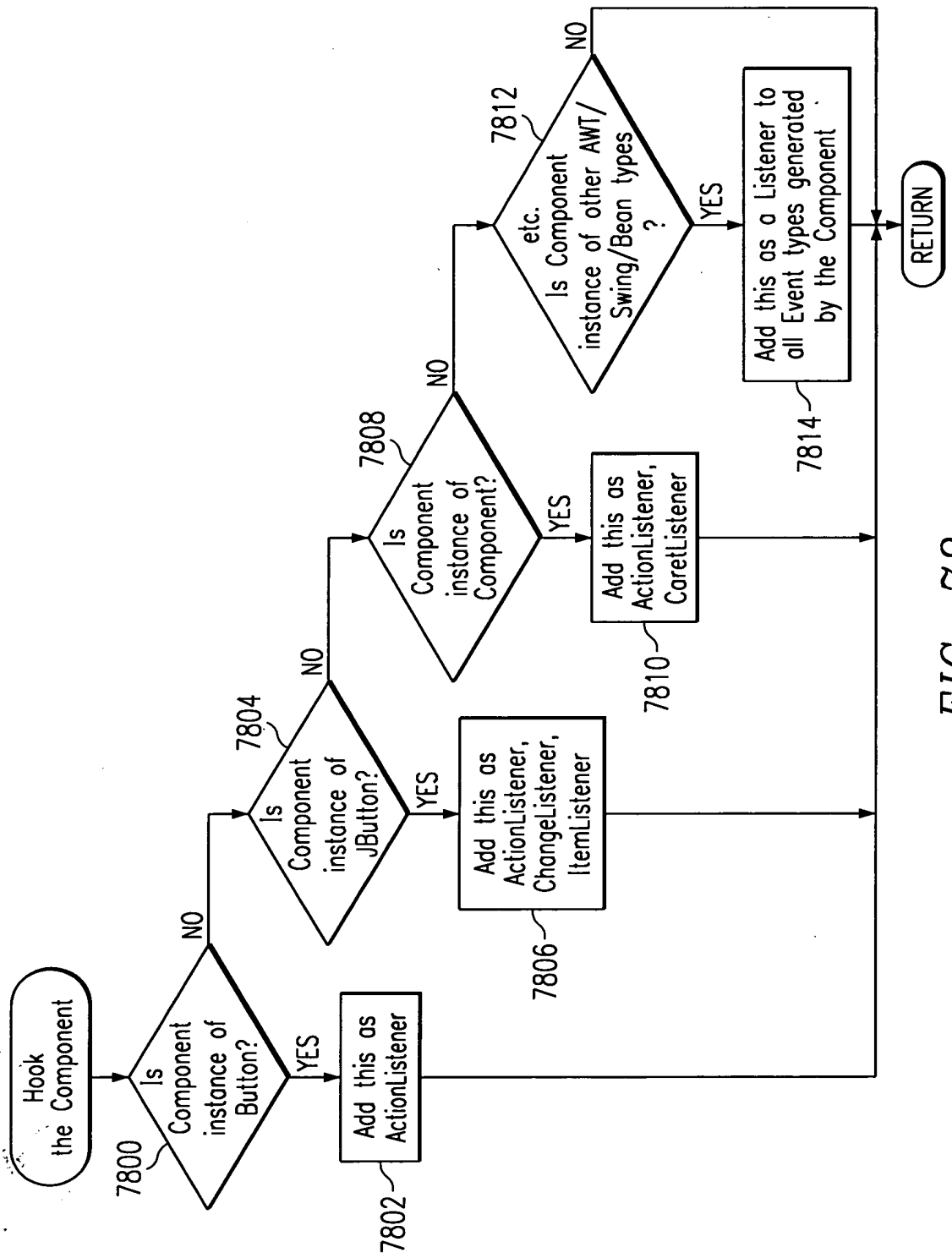

Hook the Component
7800
Is Component instance of Button?
YES
7802
Add this as ActionListener
NO
7804
Is Component instance of JButton?
YES
7806
Add this as ActionListener, ChangeListener, ItemListener
NO
7808
Is Component instance of Component?
YES
7810
Add this as ActionListener, CaretListener
NO
7812
etc. Is Component instance of other AWT/ Swing/Bean types ?
YES
7814
Add this as a Listener to all Event types generated by the Component
NO
RETURN

FIG. 78

Data Objects

✓ The ApplicationMediatorImpl will forward the refresh (default)

```
for each: ApplicationMediator -> refresh(data)
for each: ViewController -> refresh(data);
```

*FIG. 79*

✓ The ViewController will update the GUI

```
public void refresh(Object data) {
 //this example uses a keyValue pair data model

    if (data == null) return;
    else refresh((KeyValue) data);
}

public void refresh (KeyValue data) {
    nameField.setText(data.get("CustomerName"));
    idField.setText(data.get("CustomerId"));
    repaint(); //if necessary
}
```

*FIG. 80*

Data Objects

✓ How can we add a new data model (i.e. real objects)?

```
public void refresh(Object data) {
    if (data == null) return;
    else if (data instanceof Vector) {
        refresh((Vector) data);
    }
    else if (data instanceof KeyValue) {
            refresh((KeyValue) data);
        }
}
```

*FIG. 81*

```
public void refresh(Vector data) {
    //I know what they are
    Customer c = (Customer) data.elementAt(0);
    ID id = (ID) data.elementAt(1);
    nameField.setText(c.getName());
    idField.setText(id.toString());
    repaint(); //if necessary
}
```

*FIG. 82*

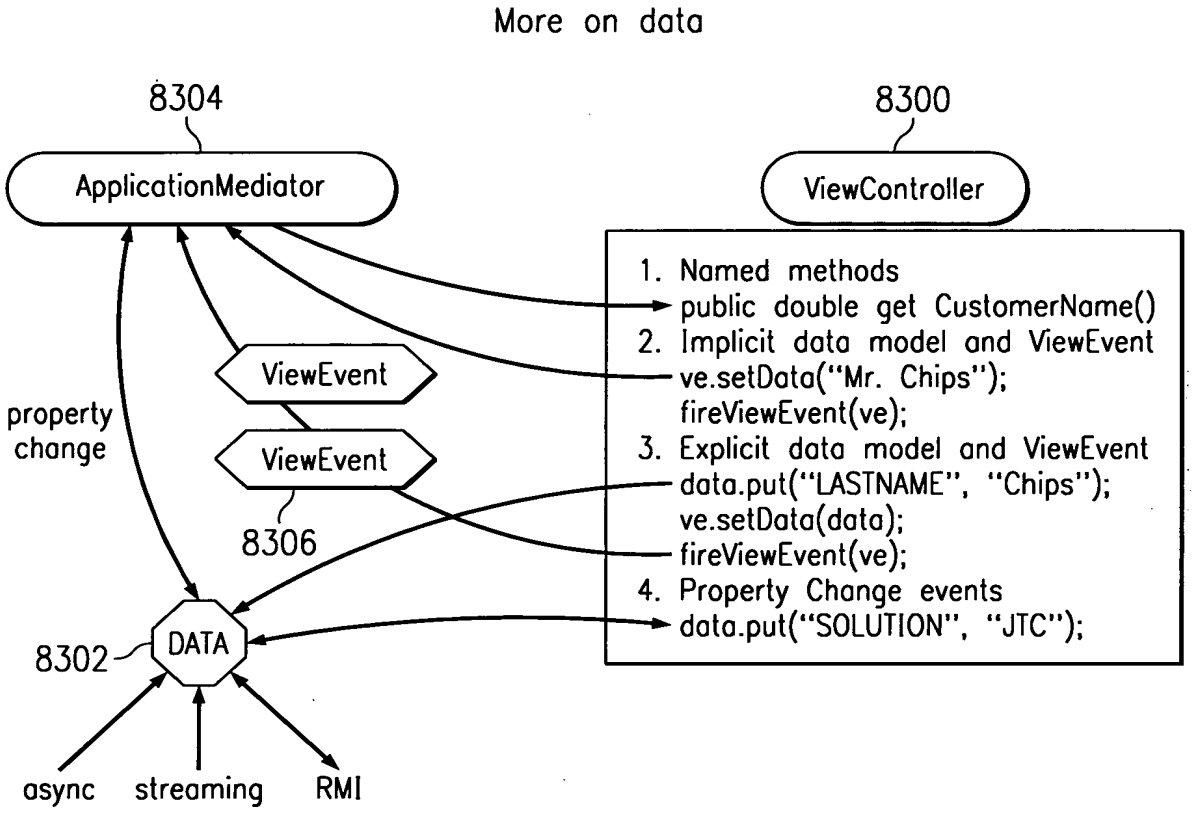

More on data
8304
ApplicationMediator
8300
ViewController
1. Named methods
public double get CustomerName()
2. Implicit data model and ViewEvent
ve.setData("Mr. Chips");
fireViewEvent(ve);
3. Explicit data model and ViewEvent
data.put("LASTNAME", "Chips");
ve.setData(data);
fireViewEvent(ve);
4. Property Change events
data.put("SOLUTION", "JTC");
ViewEvent
ViewEvent
8306
property change
8302
DATA
async
streaming
RMI

*FIG. 83*

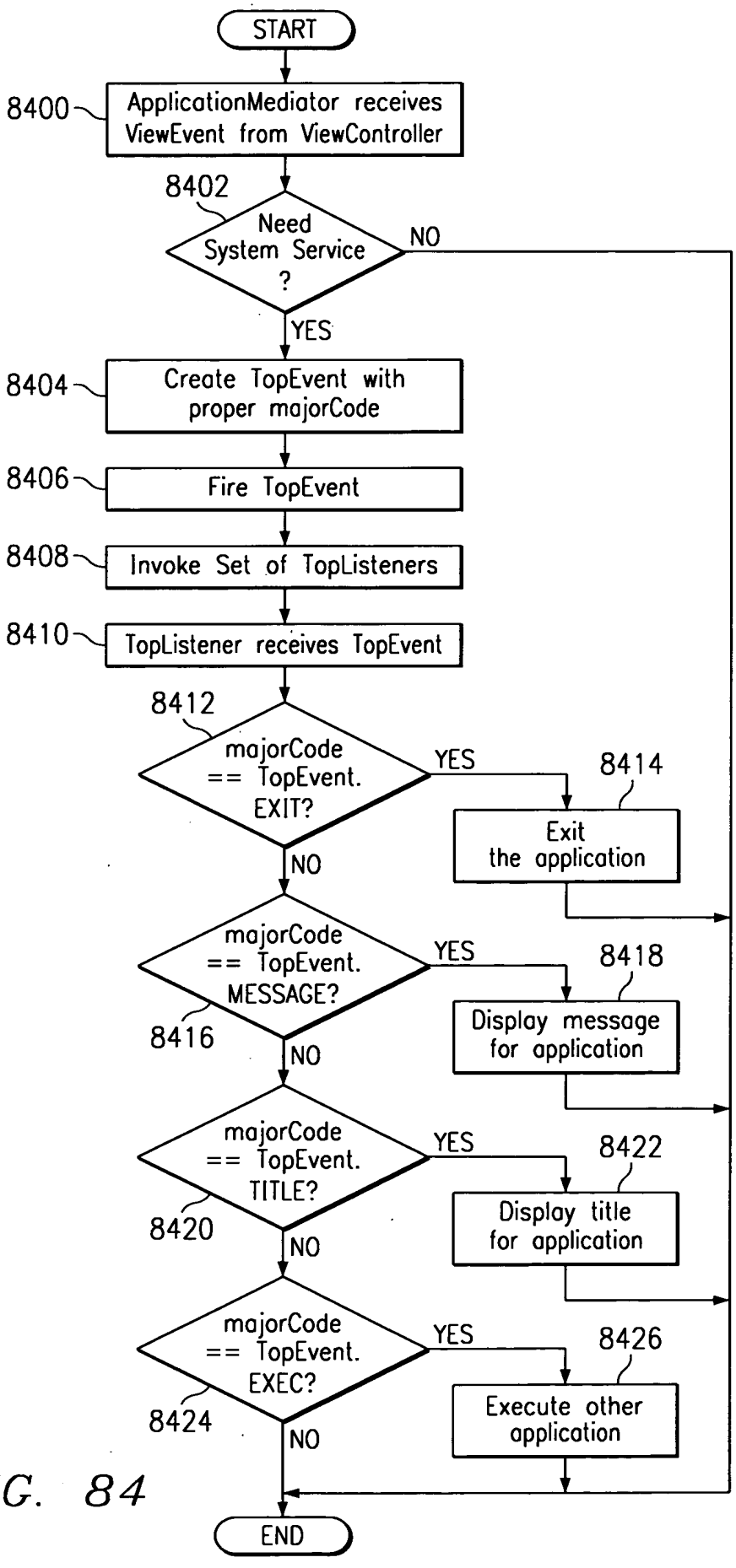

START
8400
ApplicationMediator receives ViewEvent from ViewController
8402
Need System Service ?
NO
YES
8404
Create TopEvent with proper majorCode
8406
Fire TopEvent
8408
Invoke Set of TopListeners
8410
TopListener receives TopEvent
8412
majorCode == TopEvent. EXIT?
YES
8414
Exit the application
NO
majorCode == TopEvent. MESSAGE?
YES
8418
Display message for application
8416
NO
majorCode == TopEvent. TITLE?
YES
8422
Display title for application
8420
NO
majorCode == TopEvent. EXEC?
YES
8426
Execute other application
8424
NO
END

*FIG. 84*

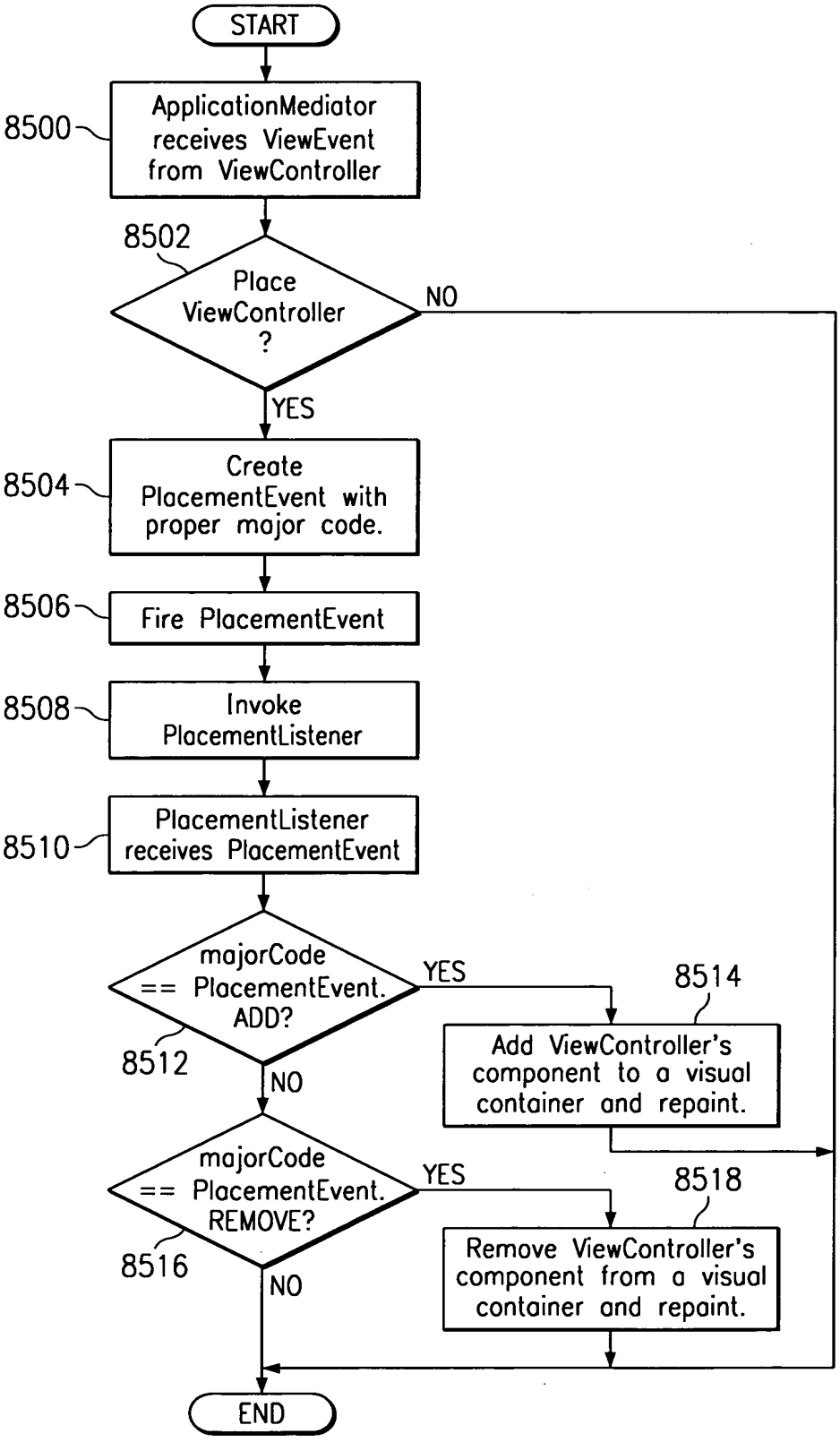

START
8500
ApplicationMediator receives ViewEvent from ViewController
8502
Place ViewController ?
NO
YES
8504
Create PlacementEvent with proper major code.
8506
Fire PlacementEvent
8508
Invoke PlacementListener
8510
PlacementListener receives PlacementEvent
majorCode == PlacementEvent. ADD?
YES
8514
Add ViewController's component to a visual container and repaint.
8512
NO
majorCode == PlacementEvent. REMOVE?
YES
8518
Remove ViewController's component from a visual container and repaint.
8516
NO
END

*FIG. 85*

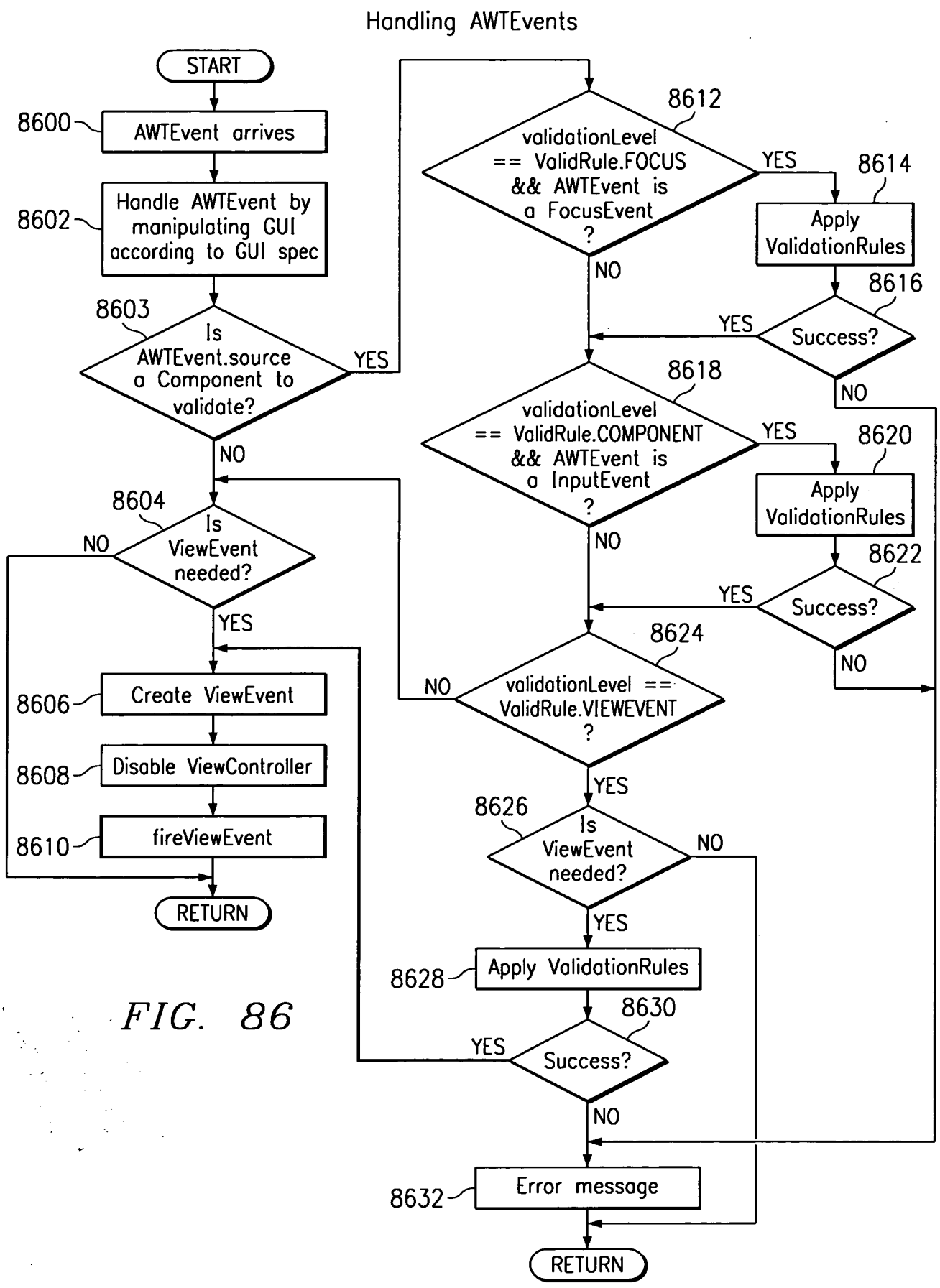

Handling AWTEvents
START
8600
AWTEvent arrives
8602
Handle AWTEvent by manipulating GUI according to GUI spec
8603
Is AWTEvent.source a Component to validate?
YES
NO
8604
Is ViewEvent needed?
NO
YES
8606
Create ViewEvent
8608
Disable ViewController
8610
fireViewEvent
RETURN
8612
validationLevel == ValidRule.FOCUS && AWTEvent is a FocusEvent ?
YES
8614
Apply ValidationRules
8616
Success?
YES
NO
NO
8618
validationLevel == ValidRule.COMPONENT && AWTEvent is a InputEvent ?
YES
8620
Apply ValidationRules
8622
Success?
YES
NO
NO
8624
validationLevel == ValidRule.VIEWEVENT ?
NO
YES
8626
Is ViewEvent needed?
NO
YES
8628
Apply ValidationRules
8630
Success?
YES
NO
8632
Error message
RETURN

*FIG. 86*

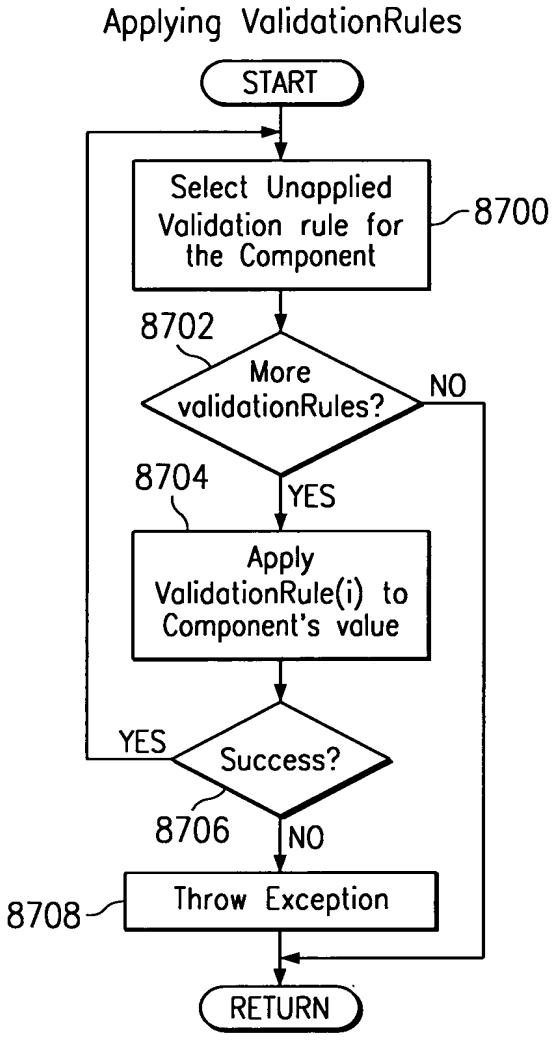



FIG. 87

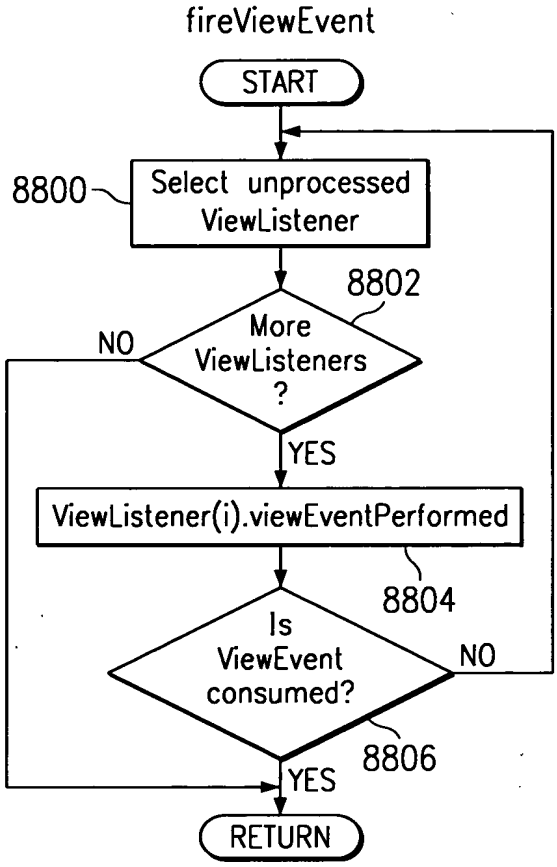



FIG. 88

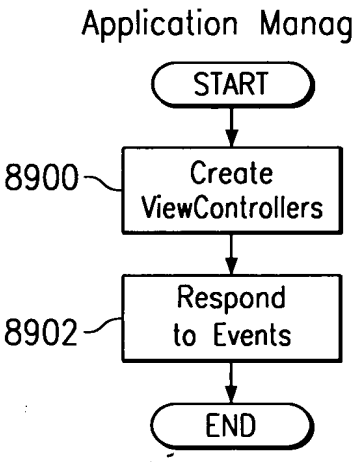



FIG. 89

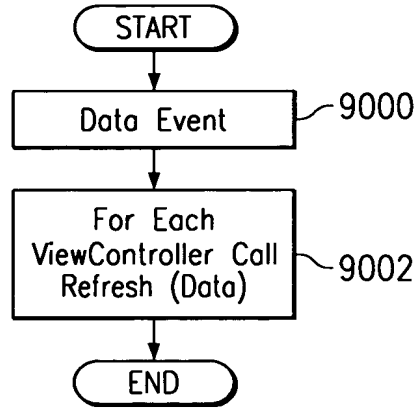



FIG. 90

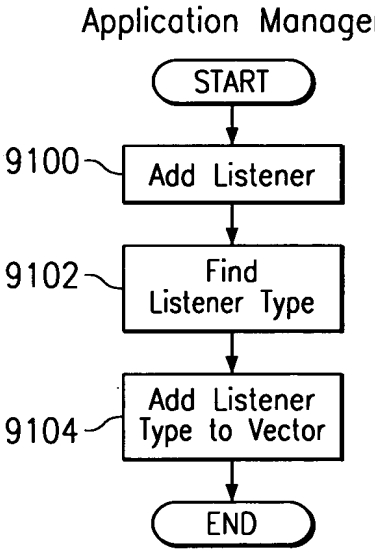



FIG. 91

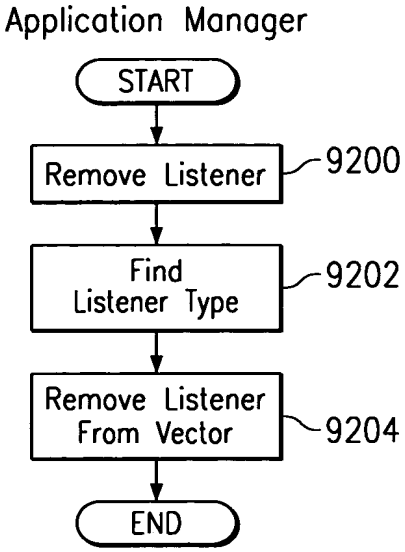



FIG. 92

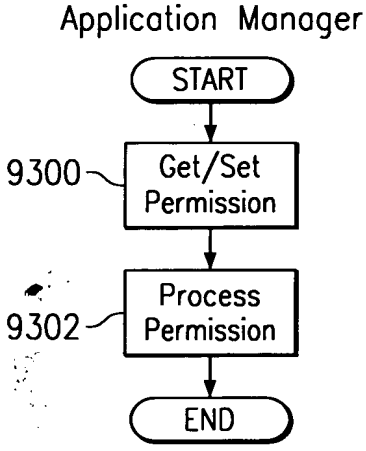



FIG. 93

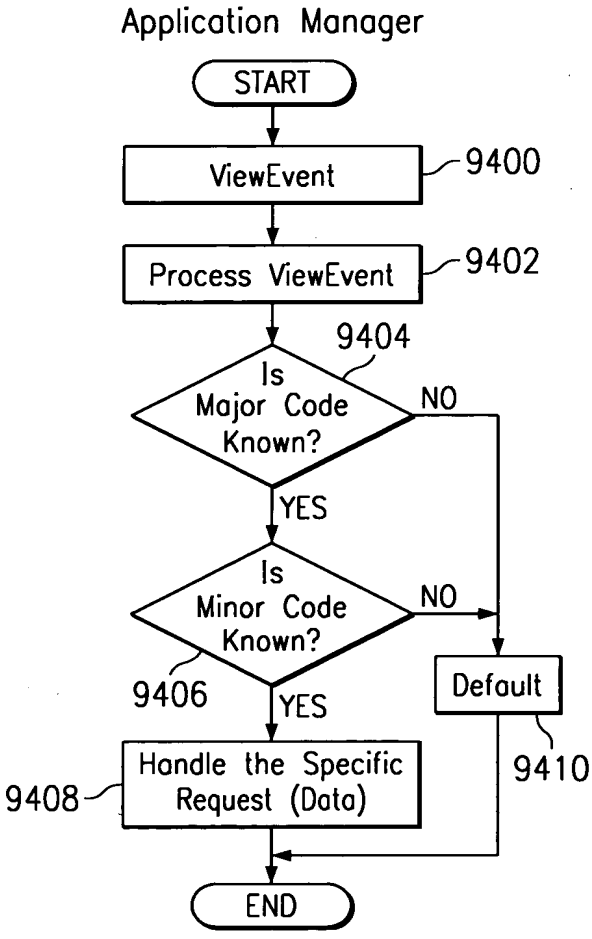



FIG. 94

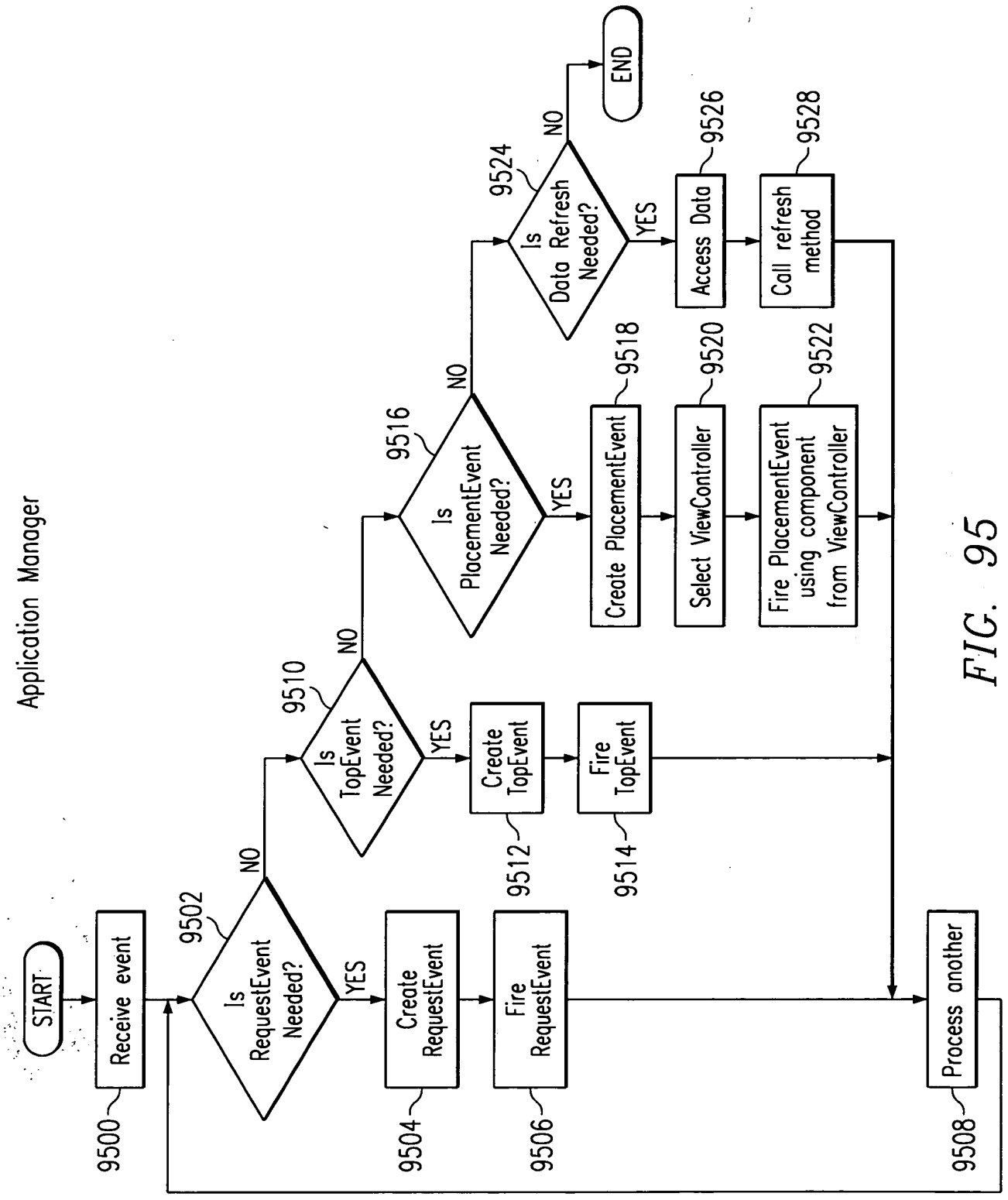

Application Manager
START
Receive event
9500
Is RequestEvent Needed?
9502
YES
NO
Create RequestEvent
9504
Fire RequestEvent
9506
Process another
9508
Is TopEvent Needed?
9510
YES
NO
Create TopEvent
9512
Fire TopEvent
9514
Is PlacementEvent Needed?
9516
YES
NO
Create PlacementEvent
9518
Select ViewController
9520
Fire PlacementEvent using component from ViewController
9522
Is Data Refresh Needed?
9524
YES
NO
Access Data
9526
Call refresh method
9528
END

FIG. 95

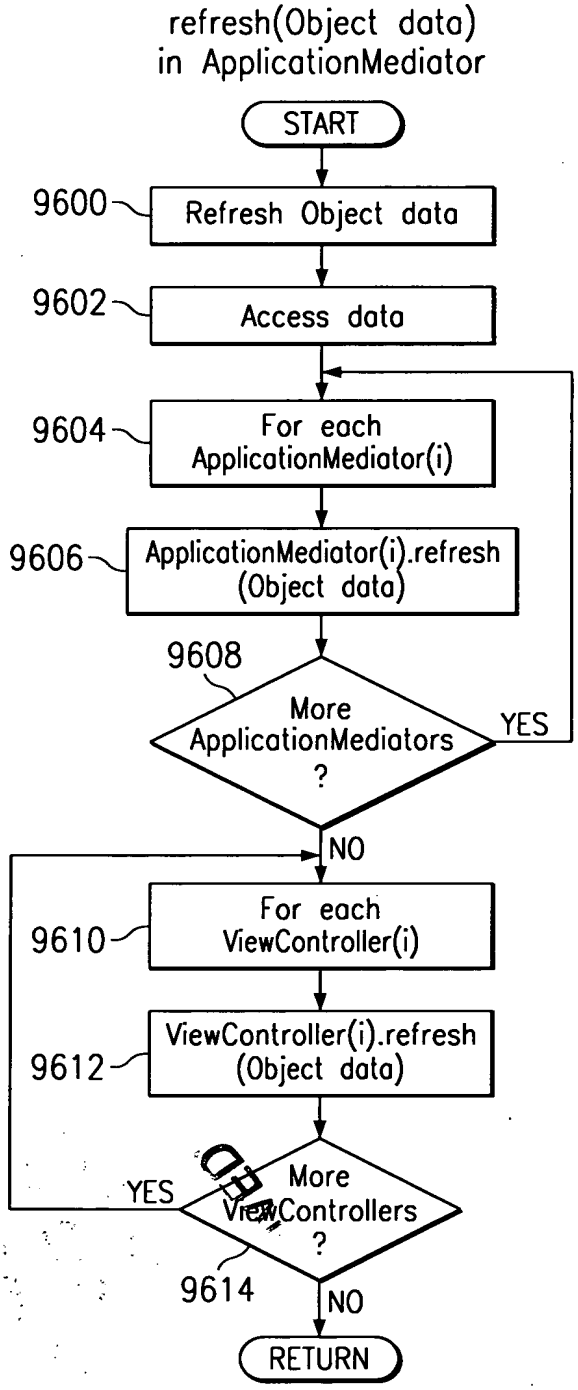

refresh(Object data)
in ApplicationMediator
START
9600
Refresh Object data
9602
Access data
9604
For each
ApplicationMediator(i)
9606
ApplicationMediator(i).refresh
(Object data)
9608
More
ApplicationMediators
?
YES
NO
9610
For each
ViewController(i)
9612
ViewController(i).refresh
(Object data)
More
ViewControllers
?
YES
9614
NO
RETURN

FIG. 96

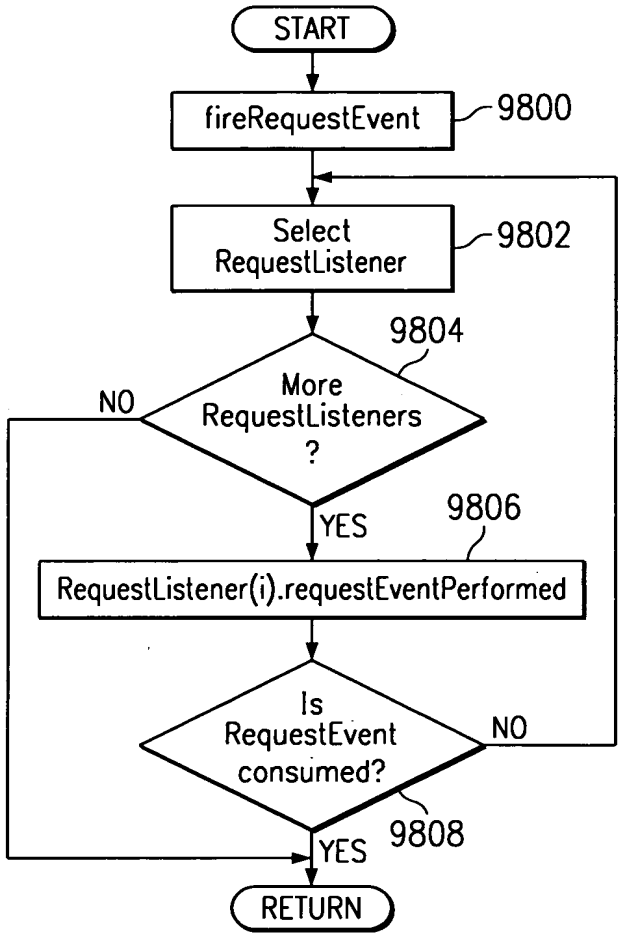

START
fireRequestEvent
9800
Select
RequestListener
9802
9804
More
RequestListeners
?
NO
YES
9806
RequestListener(i).requestEventPerformed
Is
RequestEvent
consumed?
NO
9808
YES
RETURN

FIG. 98

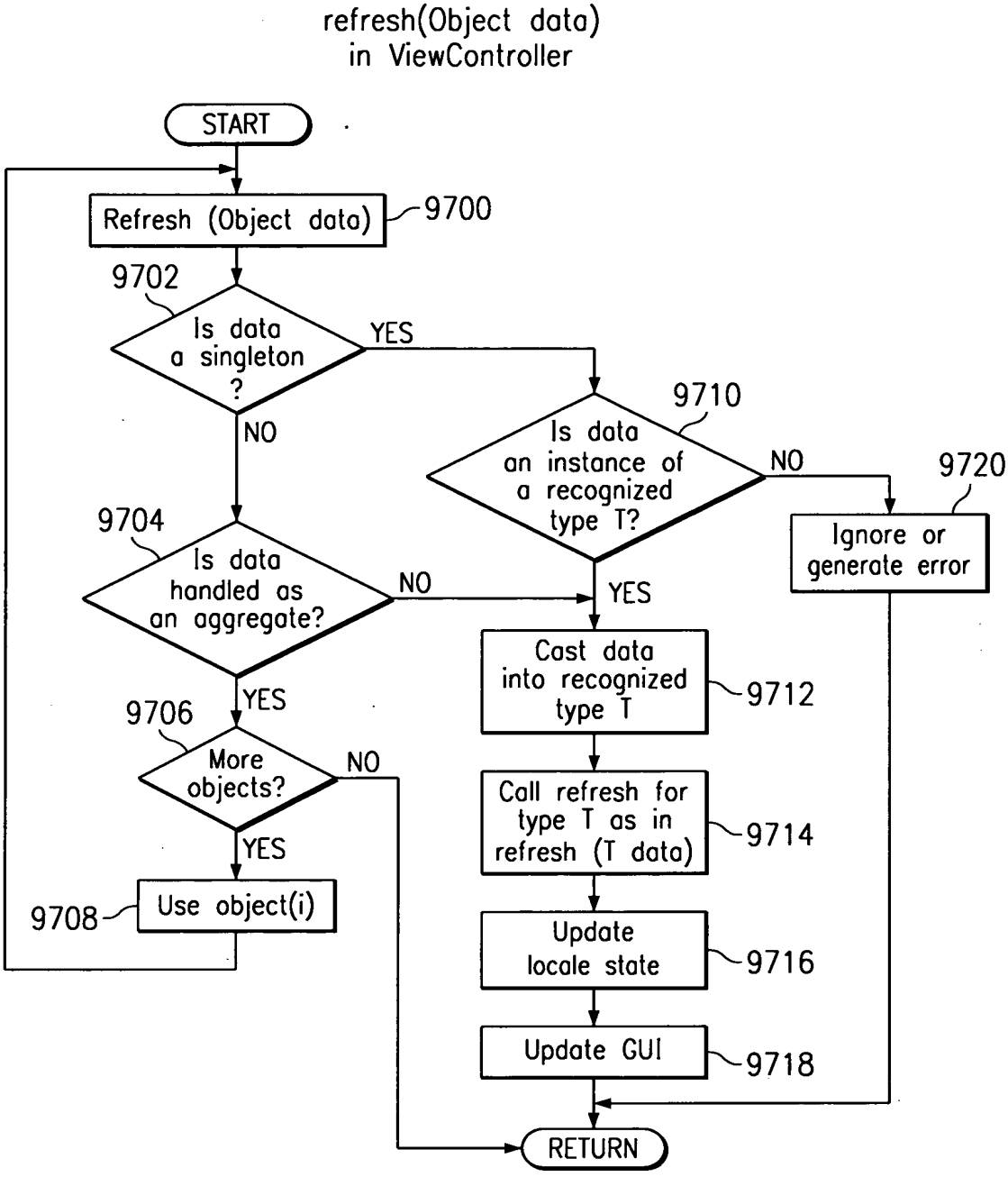

refresh(Object data)
in ViewController
START
Refresh (Object data)
9700
9702
Is data a singleton ?
YES
NO
9710
Is data an instance of a recognized type T?
NO
9720
Ignore or generate error
9704
Is data handled as an aggregate?
NO
YES
Cast data into recognized type T
9712
9706
More objects?
NO
YES
Call refresh for type T as in refresh (T data)
9714
9708
Use object(i)
Update locale state
9716
Update GUI
9718
RETURN

*FIG. 97*

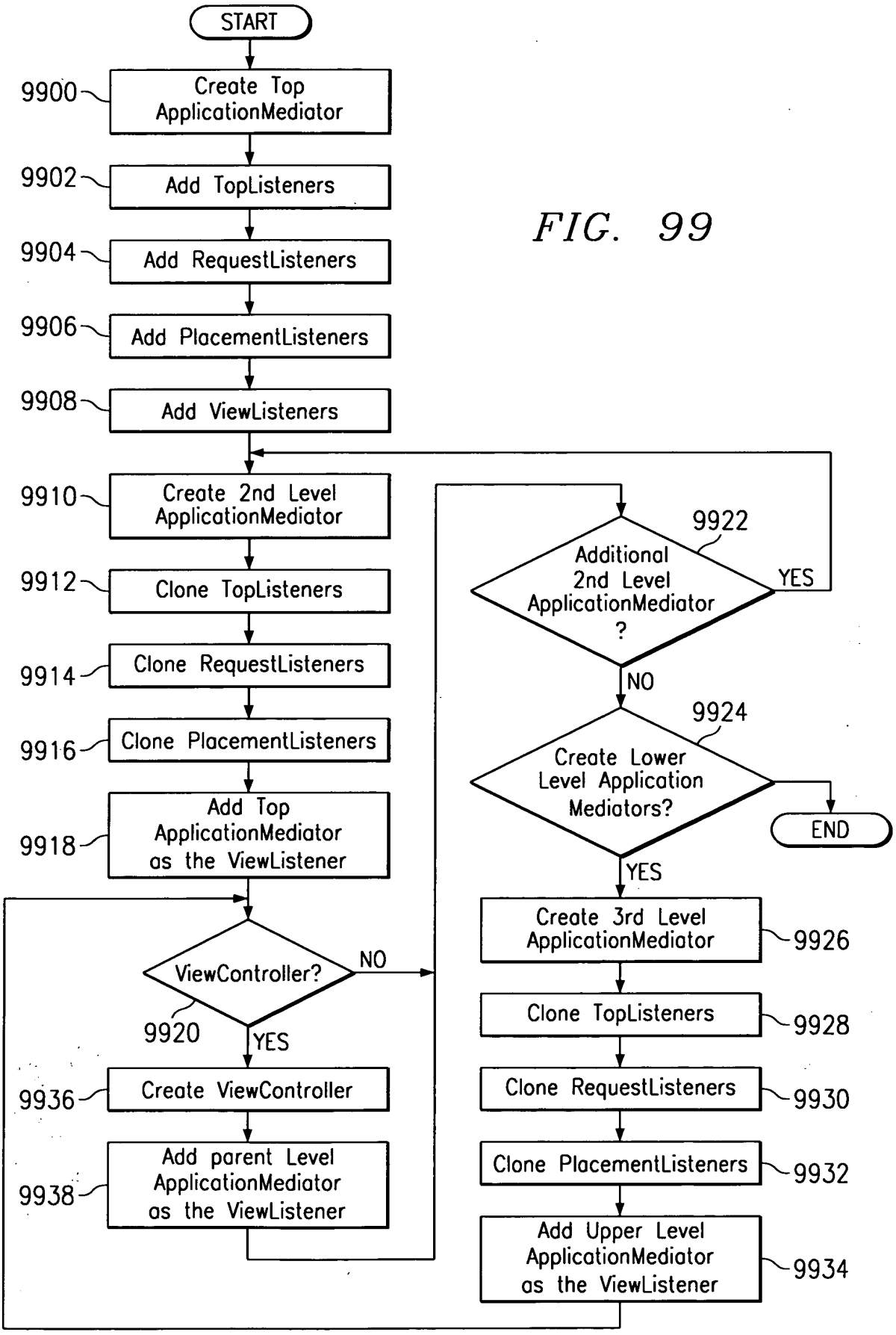

START
9900
Create Top ApplicationMediator
9902
Add TopListeners
9904
Add RequestListeners
9906
Add PlacementListeners
9908
Add ViewListeners
9910
Create 2nd Level ApplicationMediator
9912
Clone TopListeners
9914
Clone RequestListeners
9916
Clone PlacementListeners
9918
Add Top ApplicationMediator as the ViewListener
ViewController?
NO
9920
YES
9936
Create ViewController
9938
Add parent Level ApplicationMediator as the ViewListener
9922
Additional 2nd Level ApplicationMediator ?
YES
NO
9924
Create Lower Level Application Mediators?
END
YES
Create 3rd Level ApplicationMediator
9926
Clone TopListeners
9928
Clone RequestListeners
9930
Clone PlacementListeners
9932
Add Upper Level ApplicationMediator as the ViewListener
9934

*FIG. 99*

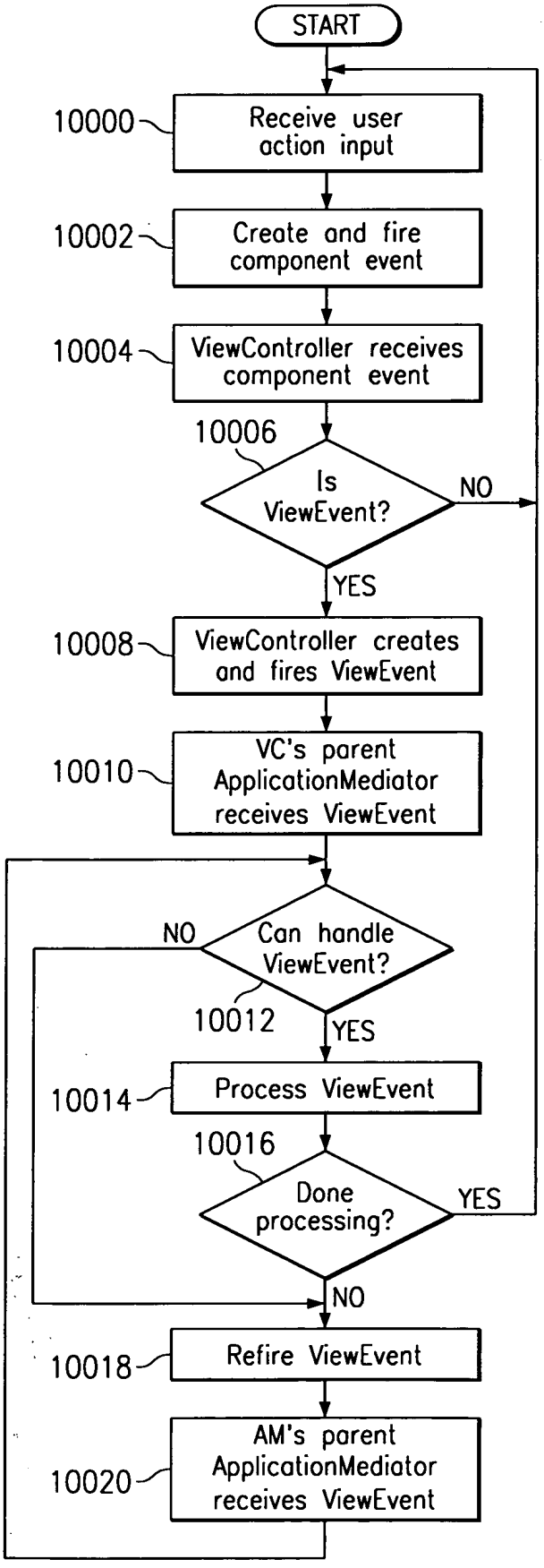



FIG. 100

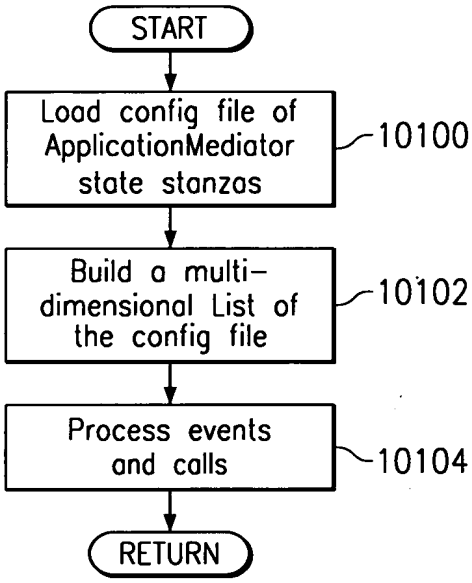



FIG. 101

Encoding ApplicationMediators

→ S1: (VE.source==vc1 && VE.major==A && VE.minor==B) =>
(RE.major=C RE.minor=D RE.data=VE.data RE.fireS)
if event source is vc1 with A,B as major/minor then
fire sync request with C,D major/minor and use data from event)

→ S2: VE.source==vc4 && VE.major==5) ==> (TE.major=3 TE.fire)
if event source is vc4 with 5 as major then
fire top event with major 3

→ S3: (Refresh) ==> (VC.i.refresh(Refresh.data))
if refresh(data) occurs, then refresh all view controllers with the same data, but not the other application mediators

→ S4: (VE.source==vcA) ==> (RE.major="set"RE.fireA) &&
(PE.major=PE.ADD PE.source=vcB PE.fire) && (VC.vcB.refresh(RE.data))
if event source is vcA, then fire async request, then fire placement event, then refresh the newly placed view controller with the data returned with the request

*FIG. 102*

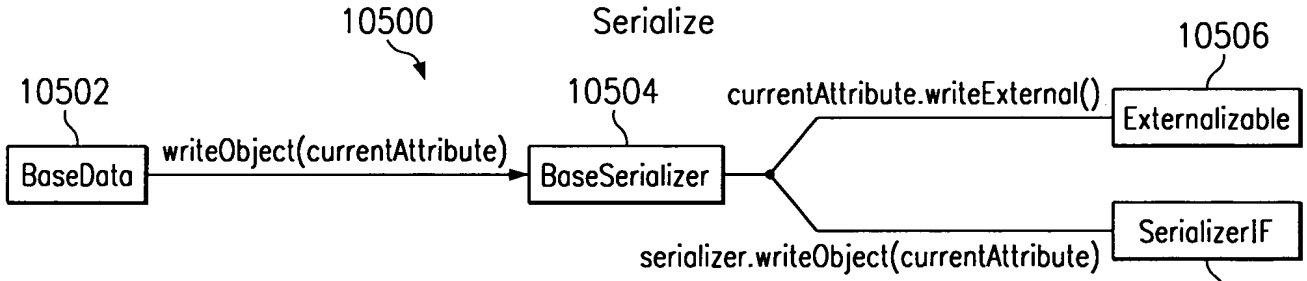



*FIG. 105*

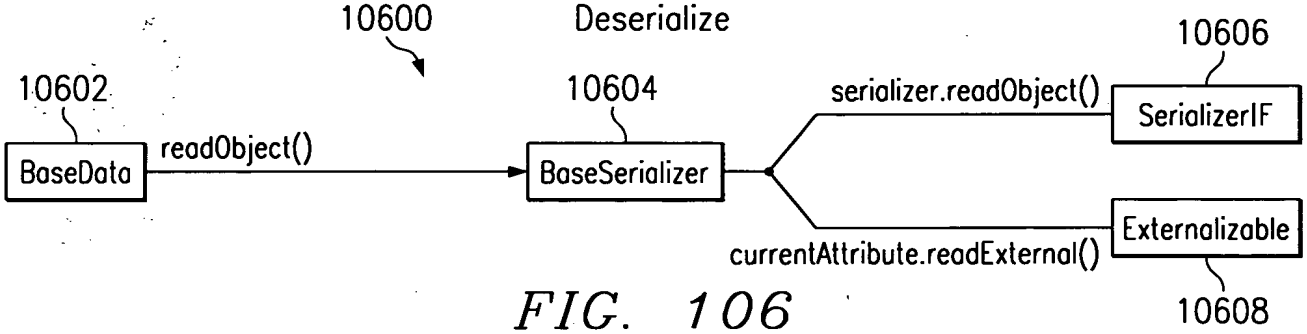



*FIG. 106*

Access State machine to see if processing is needed

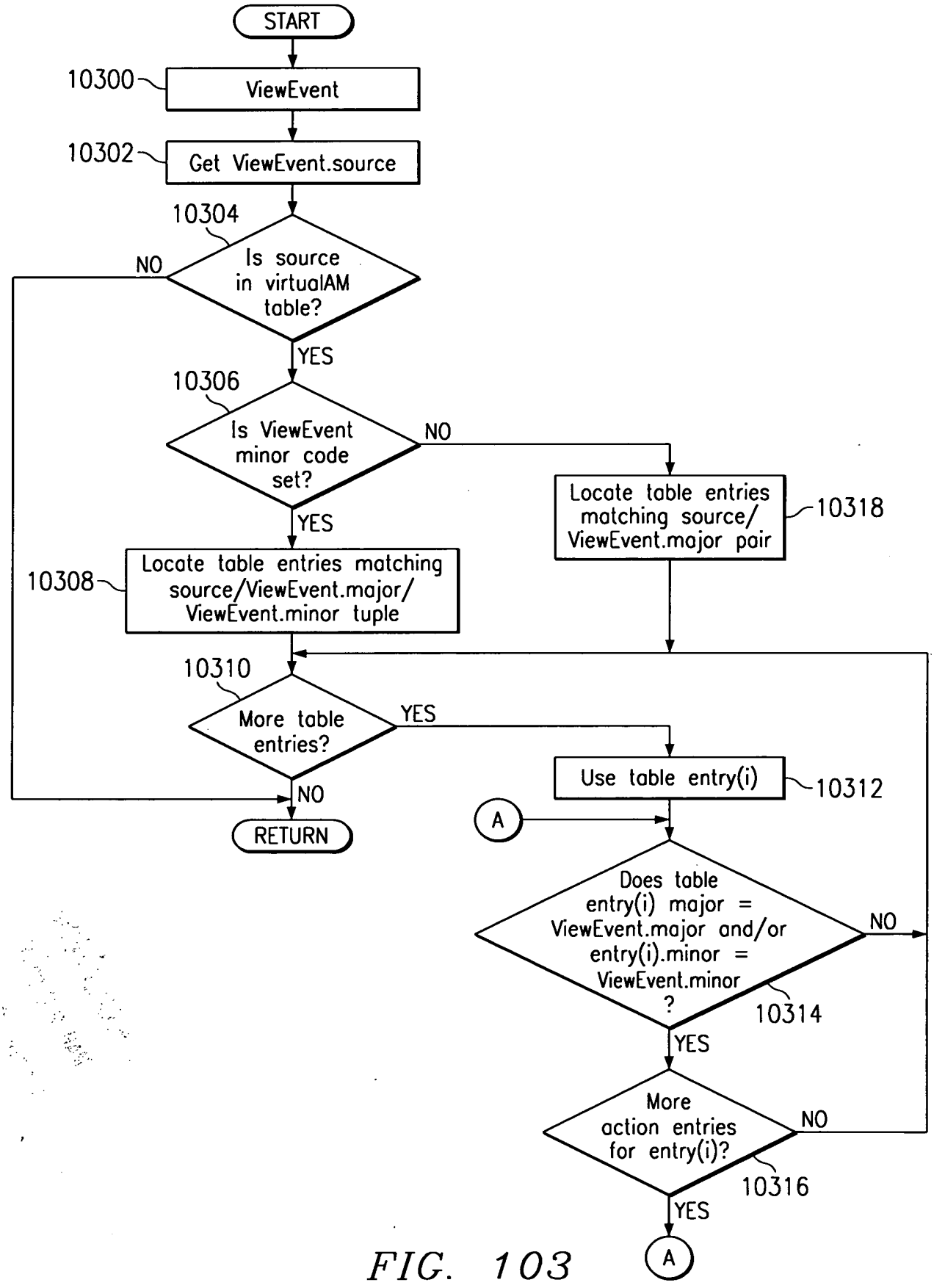



*FIG. 103*

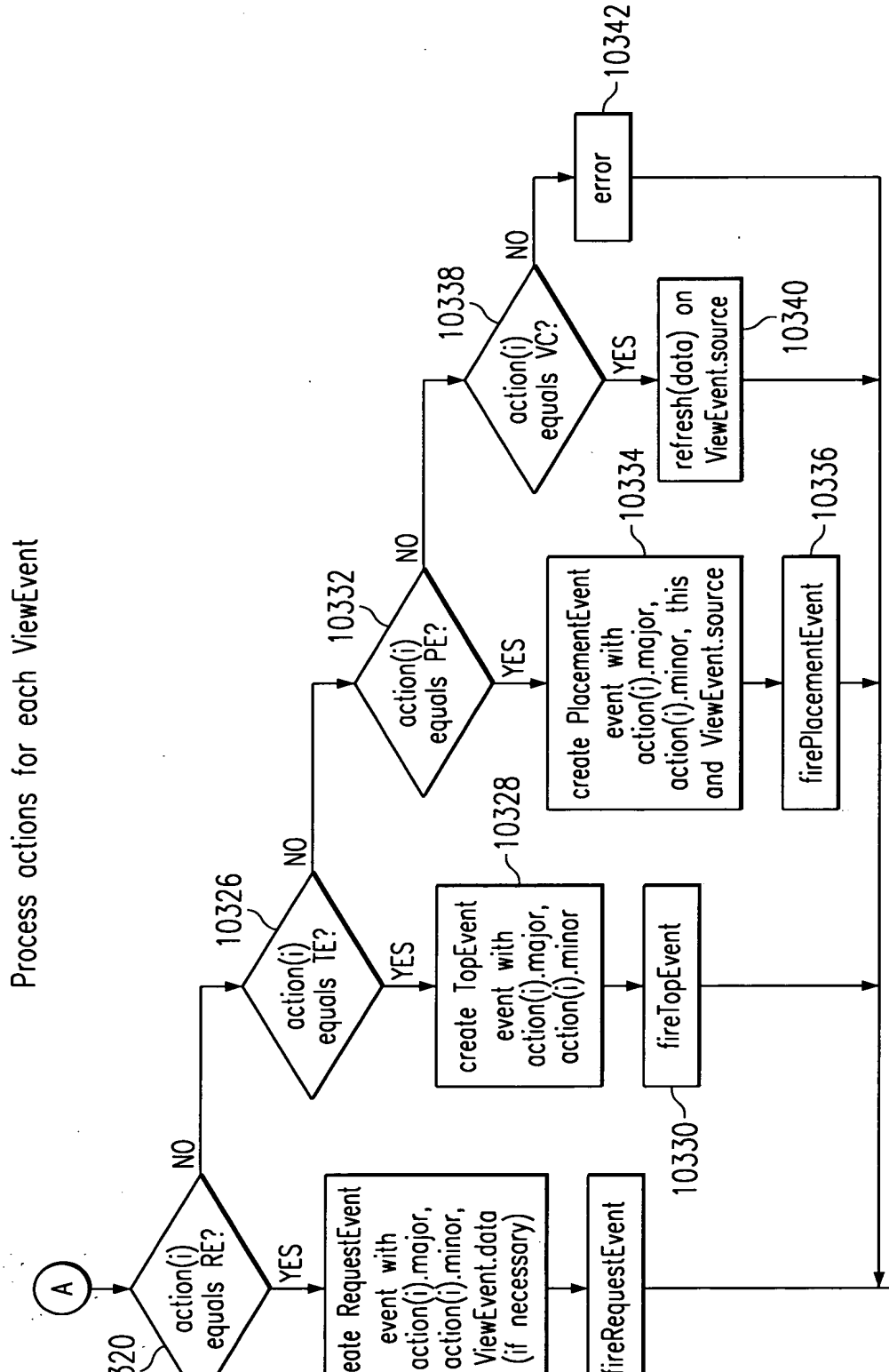

Process actions for each ViewEvent
A
10320
action(i) equals RE?
YES
NO
create RequestEvent event with action(i).major, action(i).minor, ViewEvent.data (if necessary)
10322
fireRequestEvent
10324
10326
action(i) equals TE?
YES
NO
create TopEvent event with action(i).major, action(i).minor
10328
fireTopEvent
10330
10332
action(i) equals PE?
YES
NO
create PlacementEvent event with action(i).major, action(i).minor, this and ViewEvent.source
10334
firePlacementEvent
10336
10338
action(i) equals VC?
YES
NO
refresh(data) on ViewEvent.source
10340
error
10342
RETURN

*FIG. 104*

10700

```
package com.ibm.jtcx.serialization;

import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;
/**
 * Default type comment.
 *
 * <P>INVARIANT:
 */
public class BaseData implements Externalizable {
        private Object[] data = null;
/**
 * BaseData constructor comment.
 */
public BaseData() {
        this(0);
}
/**
 * BaseData constructor comment.
 * @param dataArray java.lang.Object[]
 */
public BaseData(int count) {
        super();

        setData(new Object[count]);
}
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @return java.lang.Object[]
 */
public final Object[], getData() {
        return data;
}
```

*FIG. 107A*

10700

```
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @return java.lang.Object
 * @param index int
 */
public final Object getData(int index) {
	Object retVal = null;

	if ((data != null) && (index < data.length)) {
		retVal = data[index];
	}

	return retVal;
}
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @param in ObjectInput
 */
public void readExternal(ObjectInput in)
	throws ClassNotFoundException, IOException {

	setData((Object[])in.readObject());
}
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @param dataArray java.lang.Object[]
 */
public final void setData(Object[] dataArray) {
	data = dataArray;
}
```

*FIG. 107B*

10700

```
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @param index int
 * @param dataElement java.lang.Object
 */
public final void setData(int index, object dataElement) {
        if ((data != null) && (index < data.length)) {
                data[index] = dataElement;
        }
}
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @param out ObjectOutput
 */
public void writeExternal(ObjectOutput out) throws IOException {
        out.writeObject(getData());
}
}
```

*FIG. 107C*

10800

```
package com.ibm.jtcx.serialization;

import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.Date;
import java.util.Enumeration;
import java.util.GregorianCalendar;
import java.util.Hashtable;
import java.util.SimpleTimeZone;
import java.util.TimeZone;
import java.util.Vector;
/**
 * Base class of data objects that require small serialization. The
 * attributes of the data object are stored in an array and the elements
 * of the array are written individually.
 *
 * <P>INVARIANT:
 */
public class BaseDataS extends BaseData implements Externalizable {
/**
 * Default constructor.
 */
public BaseDataS() {
        super();
}
/**
 * Creates a new <code>BaseDataS</code> object with a data array of
 * size <code>count</code>.
 *
 * @param count the size of the data array containing the attributes
 */
public BaseDataS(int count) {
        super(count);
```

*FIG. 108A*

```
}
/**
 * Reads the array of data elements from the stream. The responsibility
 * of reading the individual element is left to the
 * <code>BaseSerializer</code> via <code>readObject()<code>.
 *
 * @param in the input stream that contains the serialized object
 * @exception ClassNotFoundException thrown if
 * <code>BaseSerializer</code> fails to read the object from the stream
 * @exception IOException thrown if
 * <code>BaseSerializer</code> fails to read the object from the stream
 * @see BaseSerializer#readObject
 */
public void readExternal(ObjectInput in)
        throws ClassNotFoundException, IOException {

        int size = in.readShort();

        if (size == -1) {
                setData(null);
        } else {
                Object[] array = new Object[size];

                for (int i = 0; i < size; i++) {
                        array[i] = BaseSerializer.getInstance().readObject(in);
                }

                setData(array);
        }
}
/**
 * Writes the array of data elements. The responsibility of writing the
 * data elements is left to <code>BaseSerializer</code> via
 * <code>writeObject()</code>.
 *
 * @param out the output stream to which the data elements will be
 * written
 */
public void writeExternal(ObjectOutput out) throws IOException {
        Object[] array = getData();

        if (array == null) {
                out.writeShort(-1);
        } else {
                out.writeShort(array.length);

                for (int i = 0; i < array.length; i++) {
                        BaseSerializer.getInstance().writeObject(out, array[i]);
                }
        }
}
}
```

10800

*FIG. 108B*

10900

```
package com.ibm.jtcx.serialization;

import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;
/**
 * The interface for those classes that serialize objects to and from
 * a stream. The object that implements this interface should write
 * just the object's attributes, not any other descriptive information
 * about the object. Typically, a <code>SerializerIF</code> knows how
 * to serialize a specific class.
 */
public interface SerializerIF {
/**
 * Reads an object from the stream.
 *
 * @return The object that was read.
 * @param in the input stream containing the object
 * @exception java.io.IOException thrown if the stream fails
 * @exception java.lang.ClassNotFoundException thrown if the stream
 * fails
 */
Object readObject(ObjectInput in) throws IOException, ClassNotFoundException;
/**
 * Writes the given object to the stream.
 *
 * @param out the output stream into which the object will be written
 * @param element the object that will be written to the stream
 * @exception java.io.IOException thrown if the stream fails
 */
void writeObject(ObjectOutput out, Object element) throws IOException;
}
```

*FIG. 109*

11000

```
package com.ibm.jtcx.serialization;

import java.io.*;
import java.math.BigInteger;
import java.math.BigDecimal;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.Hashtable;
import java.util.SimpleTimeZone;
import java.util.StringTokenizer;
import java.util.TimeZone;
import java.util.Vector;

/**
 * The <code>SerializerIF</code> that is used as the base level
 * serializer. It contains three tables used to serialize objects:
 * <br><ul>
 *          <li> codeTable: the table containing the serialization code of
 *                an object based on the name of the class
 *          <li> nameTable: the table containing the name of the class
 *                based on the serialization code
 *          <li> serializationTable: the table containing the serializer of
 *                an object based on its serialization code
 * </ul>
 * <br><br>
 * <code>BaseSerializer</code> delegates the responsibility of
 * serializing the objects to the <code>SerializerIF</code> associated
 * with that class or to the object itself if it implements
 * <code>Externalizable</code>.
 */
public class BaseSerializer implements SerializerIF {
      static private final int NULL_OBJECT = 0;
      static private final int OTHER = 0x00ff;

      static private final String HASHTABLE_SER = "ClassNameHash.ser";
      static private final String INI_FILE = "ClassNames.ini";

      static private Hashtable codeTable = null;
      static private Hashtable nameTable = null;
      static private Hashtable serializerTable = null;
      static private BaseSerializer instance = null;

      class BigDecimalSerializer implements Serializer IF {
            public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
```

*FIG. 110A*

11000

```
                int scale = in.readShort();
                int size = in.readShort();
                byte[] bytes = new byte[size];
                in.readFully(bytes);

                BigInteger temp = new BigInteger(bytes);
                return new BigDecimal(temp, scale);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
                BigDecimal bigD = (BigDecimal)element;

                int scale = bigD.scale();
                bigD.setScale(0);
                byte[] bytes = bigD.toBigInteger().toByteArray();
                bigD.setScale(scale);

                out.writeShort(scale);
                out.writeShort(bytes.length);
                out.write(bytes);
        }
}
class BigIntegerSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException
{
                int size = in.readShort();
                byte[] bytes = new byte[size];
                in.readFully(bytes);

                return new BigInteger(bytes);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
                byte[] bytes = ((BigInteger)element).toByteArray();

                out.writeShort(bytes.length);
                out.write(bytes);
        }
}
class BooleanSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException
{
                int value = in.readByte();

                return (value == 1) ? Boolean.TRUE: Boolean.FALSE;
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
                out.writeByte(((Boolean)element).booleanValue() ? 1 : 0);
        }
}
```

*FIG. 110B*

11000

```
class ByteSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            byte value = in.readByte();

            return new Byte(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeByte(((Byte)element).byteValue());
        }
    }
    class CharacterSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            char value = in.readChar();

            return new Character(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeChar(((Character)element).charValue());
        }
    }
    class DateSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException IOException {
            long value = in.readLong();

            return new Date(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeLong(((Date)element).getTime());
        }
    }
    class DoubleSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            double value = in.readDouble();

            return new Double(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeDouble(((Double)element).doubleValue());
        }
    }
```

*FIG. 110C*

11000

```
class FloatSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
                float value = in.readFloat();

                return new Float(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
                out.writeFloat(((Float)element).floatValue());
        }
}
class GregorianCalendarSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
                long time = in.readLong();
                Date date = new Date(time);
                SerializerIF serializer = BaseSerializer.getInstance();
                TimeZone tz = (TimeZone)serializer.readObject(in);

                GregorianCalender gCalender = new GregorianCalendar(tz);
                gCalendar.setTime(date);

                return gCalendar;
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
                GregorianCalendar temp = (GregorianCalendar)element;

                Date date = temp.getTime();
                TimeZone tz = temp.getTimeZone();

                out.writeLong(date.getTime());
                SerializerIF serializer = BaseSerializer.getInstance();
                serializer.writeObject(out, tz);
        }
}
class IntegerSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
                int value = in.readInt();

                return new Integer(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
                out.writeInt(((Integer)element).intValue());
        }
}
class LongSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
```

*FIG. 110D*

11000

```
            long value = in.readLong();

            return new Long(value);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeLong(((Long)element).longValue());
        }
}
class ObjectArraySerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            int size = in.readShort();

            Object[] array = new Object[size];
            for (int i = 0; i < size; i++) {
                SerializerIF serializer = BaseSerializer.getInstance();
                array[i] = serializer.readObject(in);
            }

            return array;
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            Object[] array = (Object[])element;

            out.writeShort(array.length);
            for (int i = 0; i < array length, i++) {
                SerializerIF serializer = BaseSerializer.getInstance();
                serializer.writeObject(out, array[i];
            }
        }
}
class ObjectSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            return in.readObject();
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeObject(element);
        }
}
class ShortSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            short value = in.readShort();

            return new Short(value);
        }
```

*FIG. 110E*

11000

```
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            out.writeShort(((Short)element).shortValue());
        }
}
class SimpleTimeZoneSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            int offset = in.readInt();
            SerializerIF serializer = BaseSerializer.getInstance();
            String id = (String)serializer.readObject(in);

            return new SimpleTimeZone(offset, id);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            SimpleTimeZone temp = (SimpleTimeZone)element;

            out.writeInt(temp.getRawOffset());
            SerializerIF serializer = BaseSerializer.getInstance();
            serializer.writeObject(out, temp.getID());
        }
}
class StringSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            int size = in.readShort();
            byte[] bytes = new byte[size];
            in.readFully(bytes);

            return new String(bytes);
        }
        public void writeObject(ObjectOutput out, Object element) throws IOException {
            byte[] bytes = ((String)element).getBytes();

            out.writeShort(bytes.length);
            out.write(bytes);
        }
}
class VectorSerializer implements SerializerIF {
        public Object readObject(ObjectInput in) throws ClassNotFoundException, IOException {
            int size = in.readShort();

            Vector vector = new Vector(size);
            for (int i = 0; i < size; i++) {
                SerializerIF serializer = BaseSerializer.getInstance();
                vector.addElement(serializer.readObject(in));
            }
```

*FIG. 110F*

11000

```
                return vector;
            }
            public void writeObject(ObjectOutput out, Object element) throws IOException {
                Vector temp = (Vector)element;

                Object[] array = new Object[temp.size()];
                for (int i = 0; i < array.length; i++) {
                    array[i] = temp.elementAt(i);
                }

                out.writeShort(array.length);
                for (int i = 0; i < array.length; i++) {
                    SerializerIF serializer=BaseSerializer.getInstance();
                    serializer.writeObject(out, array[i]);
                }
            }
        }
/**
 * Default constructor. The constructor is private because this is a
 * singleton class. When the object is constructed, it initializes its
 * tables.
 */
private BaseSerializer() {
        init();
}
/**
 * Adds the given elements to the three tables.
 *
 * @param className the name of the class
 * @param code the code for the given class
 * @param serializer the object responsible for serializing the given
 * class
 */
private void addDataToTables(String className, Number code, SerializerIF serializer) {
        getCodeTable().put(code, className);
        getNameTable().put(className, code);

        if (serializer != null) {
            getSerializerTable().put(code, serializer);
        }
}
```

*FIG. 110G*

11000

```
/**
 * Creates the codes and serializer objects for the default serialization
 * classes and adds them to the tables. The tables are then written to
 * a serialized file.
 */
private void createDefaultTables() {
      addDataToTables(BigDecimal.class.getName(), new Byte((byte)1), new
BigDecimalSerializer());
      addDataToTables(BigInteger.class.getName(), new Byte((byte)2), new BigIntegerSerializer());
      addDataToTables(Boolean.class.getName(), new Byte((byte)3), new BooleanSerializer());
      addDataToTables(Byte.class.getName(), new Byte((byte)4), new ByteSerializer());
      addDataToTables(Character.class.getName(), new Byte((byte)5), new CharacterSerializer());
      addDataToTables(Date.class.getName(), new Byte((byte)6), new DateSerializer());
      addDataToTables(Double.class.getName(), new Byte((byte)7), new DoubleSerializer());
      addDataToTables(Float.class.getName(), new Byte((byte)8), new FloatSerializer());
      addDataToTables(GregorianCalendar.class.getName(), new Byte((byte)9), new
GregorianCalendarSerializer();
      addDataToTables(Integer.class.getName(), new Byte((byte)10), new IntegerSerializer());
      addDataToTables(Long.class.getName(), new Byte((byte)11), new LongSerializer());
      addDataToTables(Short.class.getName(), new Byte((byte)12), new ShortSerializer());
      addDataToTables(SimpleTimeZone.class.getName(), new Byte((byte)13), new
SimpleTimeZoneSerializer());
      addDataToTables(String.class.getName(), new Byte((byte)14), new StringSerializer());
      addDataToTables(Vector.class.getName(), new Byte((byte)15), new VectorSerializer());
      addDataToTables(Object.class.getName(), new Byte((byte)16), new ObjectSerializer());

      writeTables();
}
/**
 * Returns an instance of the table of class names, keyed by their code.
 * If the table does not exist, it is created.
 *
 * @return The table of class names.
 */
protected Hashtable getCodeTable() {
      if (codeTable == null) {
            codeTable = new Hashtable();
      }
```

*FIG. 110H*

11000

```
        return codeTable;
}
/**
* Returns an instance of <code>BaseSerializer</code>.
*
* @return An instance of <code>BaseSerializer</code>.
*/
public static SerializerIF getInstance() {
        if (instance == null) {
                instance = new BaseSerializer();
        }

        return instance;
}
/**
* Returns an instance of the table of codes, keyed by their
* corresponding class name.
* If the table does not exist, it is created.
*
* @return The table of codes.
*/
protected Hashtable getNameTable() {
        if (nameTable == null) {
                nameTable = new Hashtable();
        }

        return nameTable;
}
/**
* Returns an instance of the table of serializers, keyed by their
* corresponding code.
* If the table does not exist, it is created.
*
* @ return The table of class names.
*/
protected Hashtable getSerializerTable() {
        if (serializerTable == null) {
                serializerTable = new Hashtable();
        }

        return serializerTable;
}
/**
* Initializes the hashtable from either a serialized hashtable or from
* an ini file.
*/
```

*FIG. 110I*

11000

```
protected void init() {
        File serializedFile = new File(HASHTABLE_SER);
        File iniFile = new File(INI_FILE);

        if (serializedFile.exists()) {
                readSerializedFile(serializedFile);
        } else {
                if (iniFile.exists()) {
                        readIniFile(iniFile);
                }

                createDefaultTables();
        }
}
/**
 * Gets the value of the serialization code from the table based on
 * the className provided. The value returned can either be a
 * <code>Byte</code> or an <code>Integer</code>. The return value
 * will be a <code>Byte</code> if the className is one of the base
 * data types.
 *
 * @return The serialization code of the className.
 * @param className the name of the class
 */
private Number lookupCode(String className) {
        Number code = null;

        if (className != null) {
                code = (Number)getNameTable().get(className);
        }

        return code;
}
/**
 * Looks up the hashcode in the table and returns the String value of
 * the hashcode. If the hashcode does not exist in the table
 * <code>null</code> is returned.
 *
 * @return The object that was stored in the table with the given
 *                hashcode.
 * @param hashcode the hashcode that will be used to look up the value
 */
```

*FIG. 110J*

11000

```
private String lookupName(Number code) {
        String className = null;

        if (code != null) {
                className = (String)getCodeTable().get(code);
        }

        return className;
}
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @return com.ibm.jtc.util.SerializerIF
 * @param code int
 */
private SerializerIF lookupSerializer(Number code) {
        SerializerIF serializer = null;

        if (code != null) {
                serializer = (SerializerIF)getSerializerTable().get(code);
        }

        return serializer;
}
/**
 * Default method comment.
 *
 * <P>PRE:
 * <P>POST:
 *
 * @return Parameter not modified
 * @param iniFile java.io.File
 */
private void readIniFile(File iniFile) {
        BufferedReader in = null;

        try {
                in = new BufferedReader(new FileReader(iniFile));

                for (String inLine = in.readLine(); inLine != null; inLine = in.readLine()) {
                        String trimLine = inLine.trim();
```

*FIG. 110K*

11000

```
                if ((trimLine.length() > 0) &&
                        !trimLine.startsWith("#")) {
                        StringTokenizer tokenizer = new StringTokenizer(trimLine);

                        String className = tokenizer.nextToken();
                        Integer code = new Integer(className.hashCode());
                        SerializerIF serializer = null;

                        if (tokenizer.hasMoreTokens()) {
                            String serializerName = tokenizer.nextToken();

                            try {
                                serializer = (SerializerIF)Class.forName(serializerName).newInstance();
                            } catch(Exception e) { }
                        }

                        addDataToTables(className, code, serializer);
                }
            }
        } catch (Exception throwAway) {
        } finally {
            try {
                in.close();
            } catch (Exception throwAway) {
            }
        }

        writeTables();
    }
    /**
    * Reads the object from the stream by first reading the code for the
    * element then reads the appropriate data for that object.
    *
    * @return The object that was read from the stream.
    * @param in the input stream that contains the object
    */
    public Object readObject(ObjectInput in)
        throws ClassNotFoundException, IOException {
        Object retVal = null;
        Number code = null;

        byte baseCode = in.readByte();
```

*FIG. 110L*

11000

```
        if (baseCode == NULL_OBJECT) {
                retVal = null;
        } else {
                if (baseCode != OTHER) {
                        code = new Byte(baseCode);
                } else {
                        int secondCode = in.readInt();
                        code = new Integer(secondCode);
                }

                SerializerIF serializer = lookupSerializer(code);
                if (serializer != null) {
                        retVal = serializer.readObject(in);
                } else {
                        String className = lookupName(code);

                        try {
                                retVal = Class.forName(className).newInstance();

                                if (retVal instanceof Externalizable) {
                                        ((Externalizable)retVal).readExternal(in);
                                } else {
                                        retVal = in.readObject();
                                }
                        } catch(Exception e) {
                        }
                }
        }

        return retVal;
}
/**
 * Reads the file containing the serialized hashtables of data.
 *
 * @param serializedFile the file containing the serialized tables
 */
private void readSerializedFile(File serializedFile) {
        ObjectInputStream in = null;
        try {
                in = new ObjectInputStream(new FileInputStream(serializedFile));
                codeTable = (Hashtable)in.readObject();
                nameTable = (Hashtable)in.readObject();
                serializerTable = (Hashtable)in.readObject();
```

*FIG. 110M*

11000

```
        } catch (Exception throwAway) {
        } finally {
                try {
                        in.close();
                } catch (Exception throwAway) { }

                if ((codeTable == null) ||
                        (nameTable == null) ||
                        (serializerTable == null)) {

                        createDefaultTables();
                }
        }
}
/**
 * Writes the given object to the stream. First, the code representing
 * the type of the object is written, then the data within the object
 * is written.
 *
 * @param out the output stream that will contain the object
 * @param element the data object that will be written
 */
public void writeObject(ObjectOutput out, Object element)
        throws IOException {

        if (element == null) {
                out.writeByte(NULL_OBJECT);
        } else {
                String className = element.getClass().getName();
                Number code = lookupCode(className);

                if (code != null) {
                        if (code instanceof Byte) {
                                out.writeByte(code.byteValue());
                        } else if (code instanceof Integer) {
                                out.writeByte(OTHER);
                                out.writeInt(code.intValue());
                        }

                        SerializerIF serializer = lookupSerializer(code);

                        if (serializer != null) {
                                serializer.writeObject(out, element);
                        } else if (element instanceof Externalizable) {
                                ((Externalizable)element).writeExternal(out);
```

FIG. 110N

11000

```
                } else {
                        out.writeObject(element);
                }
        } else {
                if (element instanceof Object[]) {
                        className = Object[].class.getName();
                } else {
                        className = Object.class.getName();
                }

                code = lookupCode(className);
                SerializerIF serializer = lookupSerializer(code);

                out.writeByte(code.byteValue());
                serializer.writeObject(out, element);
        }
    }
}
/**
 * Writes the tables to the file.
 */
private void writeTables() {
    ObjectOutputStream out = null;

    try {
        File serFile = new File(HASHTABLE_SER);
        out = new ObjectOutputStream(new FileOutputStream(serFile));

        out.writeObject(getCodeTable());
        out.writeObject(getNameTable());
        out.writeObject(getSerializerTable());
        out.writeObject(new Date());
    } catch(Exception e) {
    } finally {
        try {
            out.close();
        } catch(Exception e) { }
    }
}
}
```

*FIG. 1100*

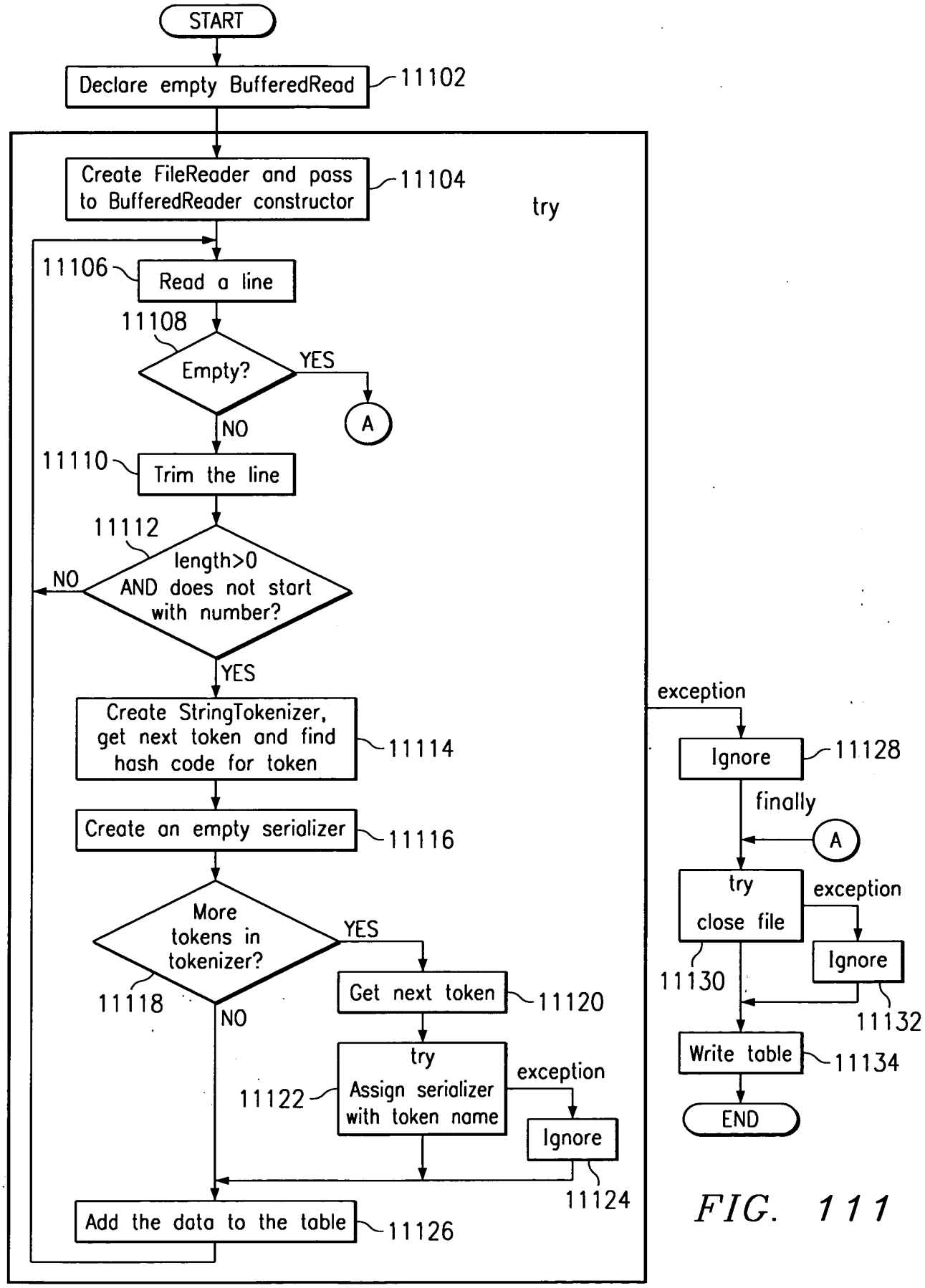

START
Declare empty BufferedRead
11102
Create FileReader and pass to BufferedReader constructor
11104
try
11106
Read a line
11108
Empty?
YES
A
NO
11110
Trim the line
11112
length>0 AND does not start with number?
NO
YES
Create StringTokenizer, get next token and find hash code for token
11114
Create an empty serializer
11116
More tokens in tokenizer?
YES
11118
NO
Get next token
11120
try
Assign serializer with token name
11122
exception
Ignore
11124
Add the data to the table
11126
exception
Ignore
11128
finally
A
try
close file
exception
11130
Ignore
11132
Write table
11134
END

*FIG. 111*

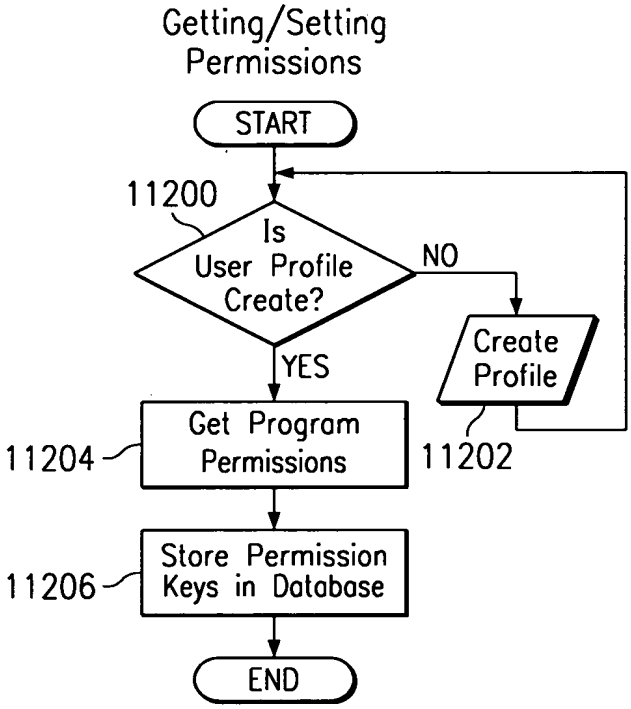



FIG. 112

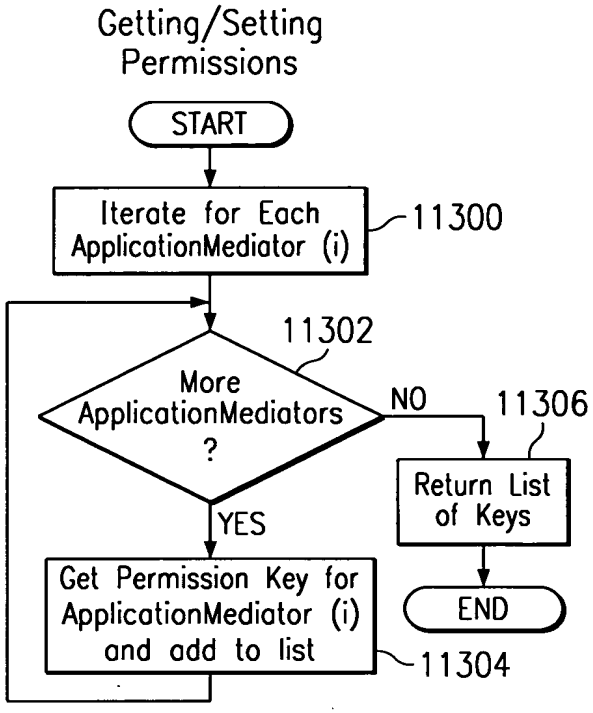



FIG. 113

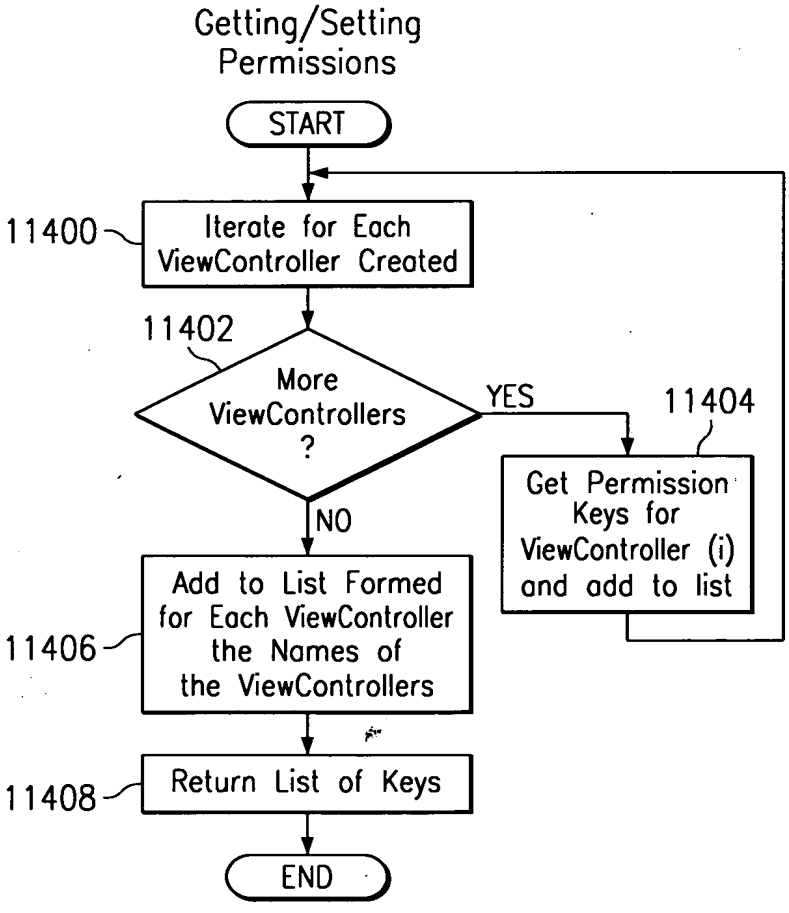



FIG. 114

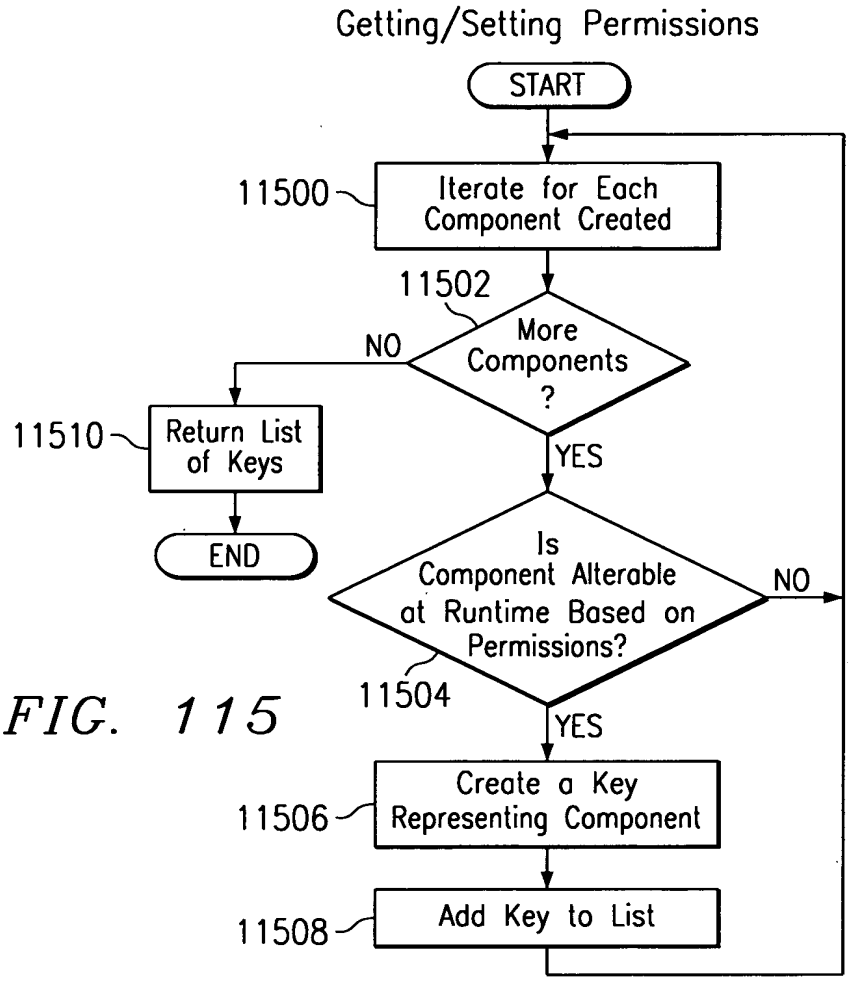



FIG. 115

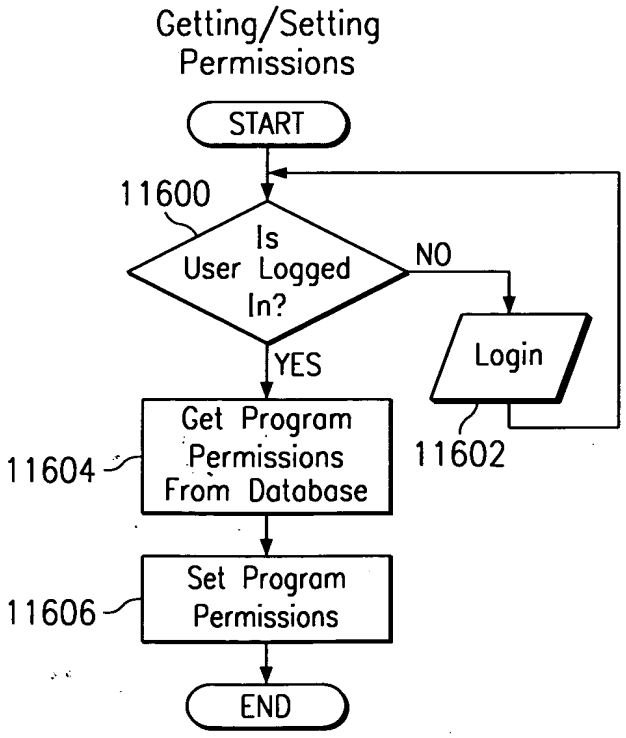



FIG. 116

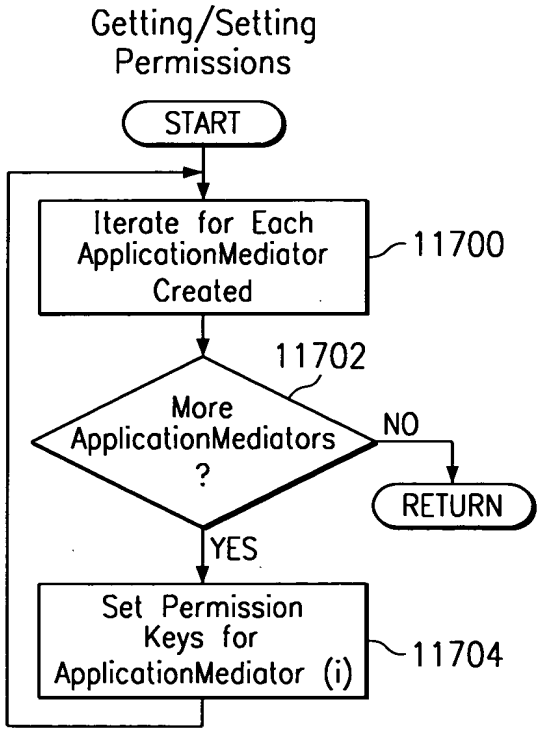



FIG. 117

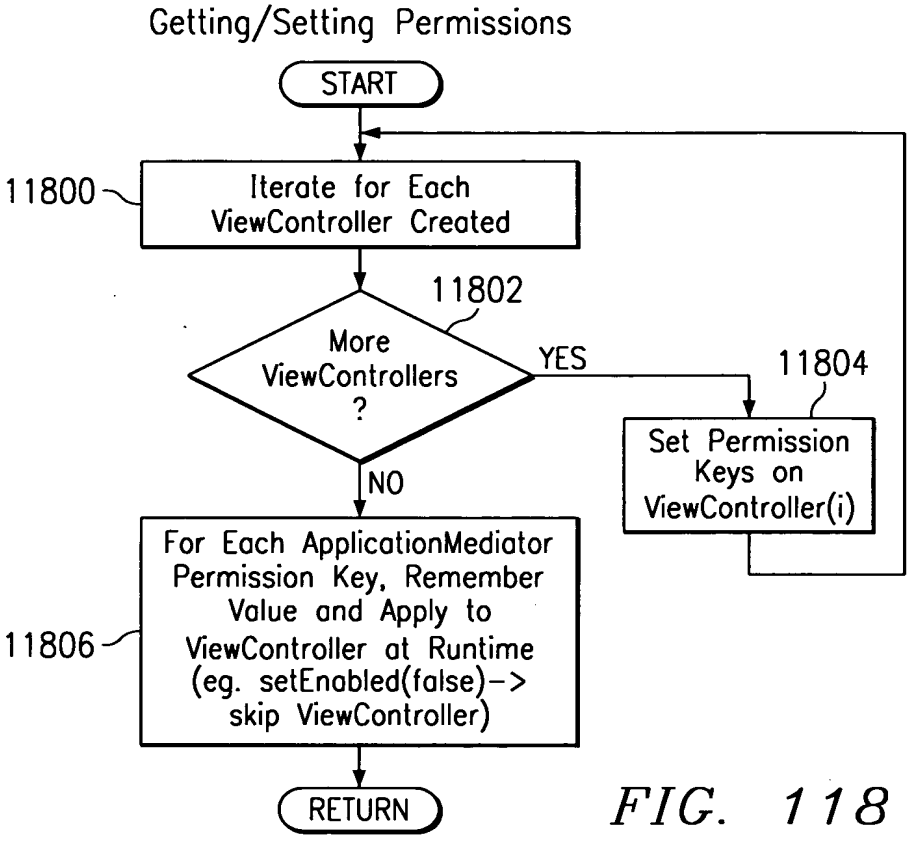



*FIG. 118*

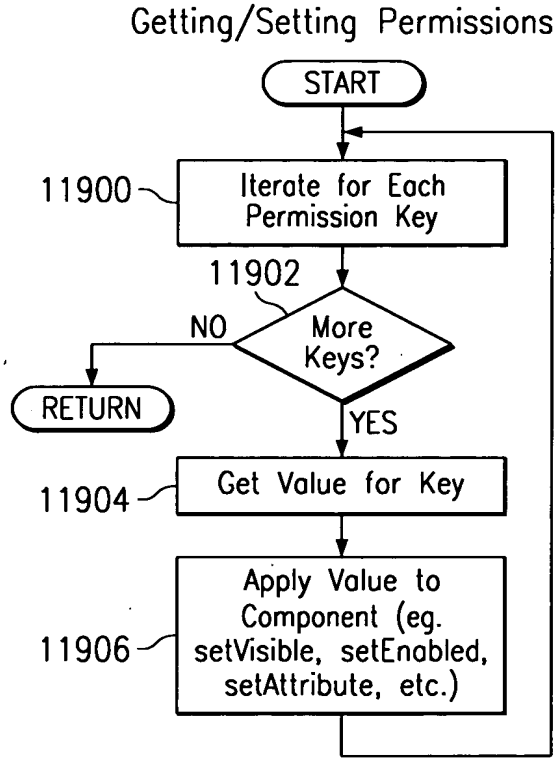



*FIG. 119*

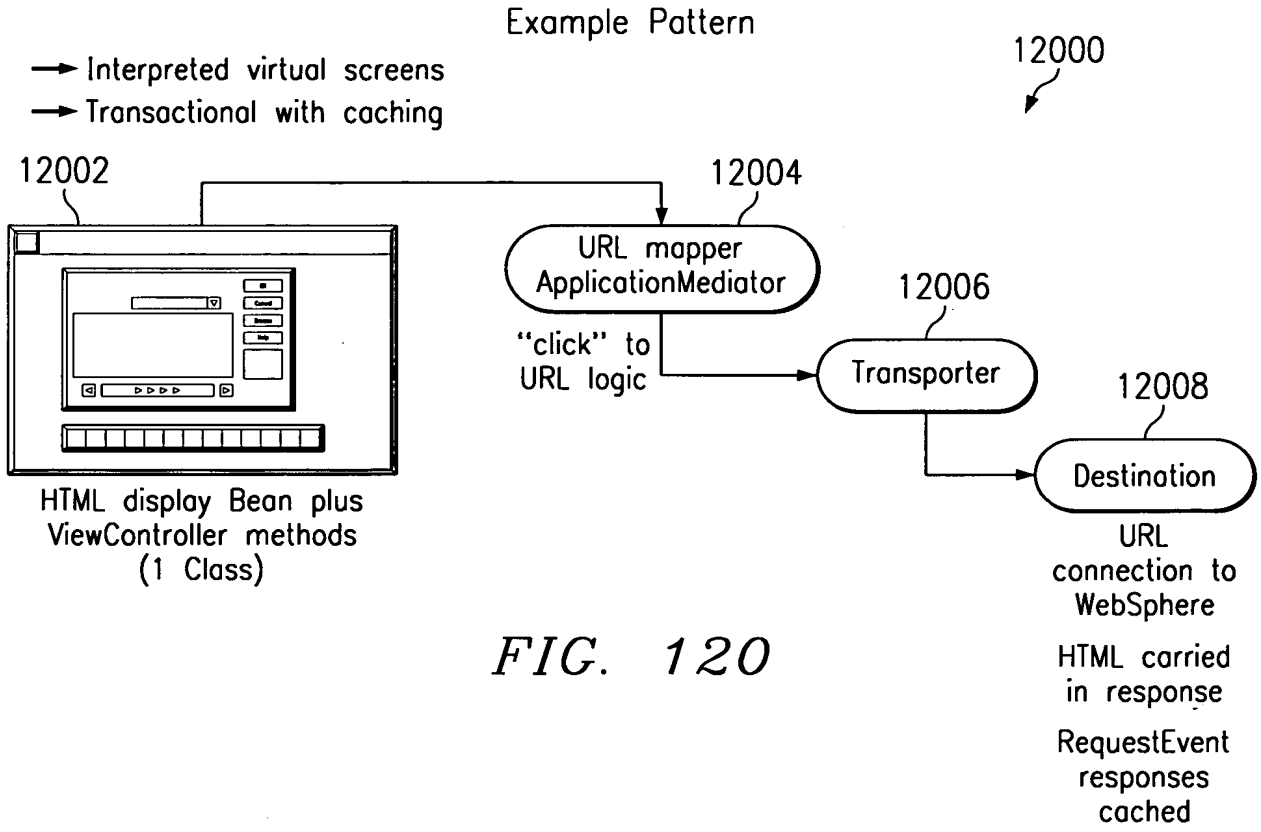



*FIG. 120*

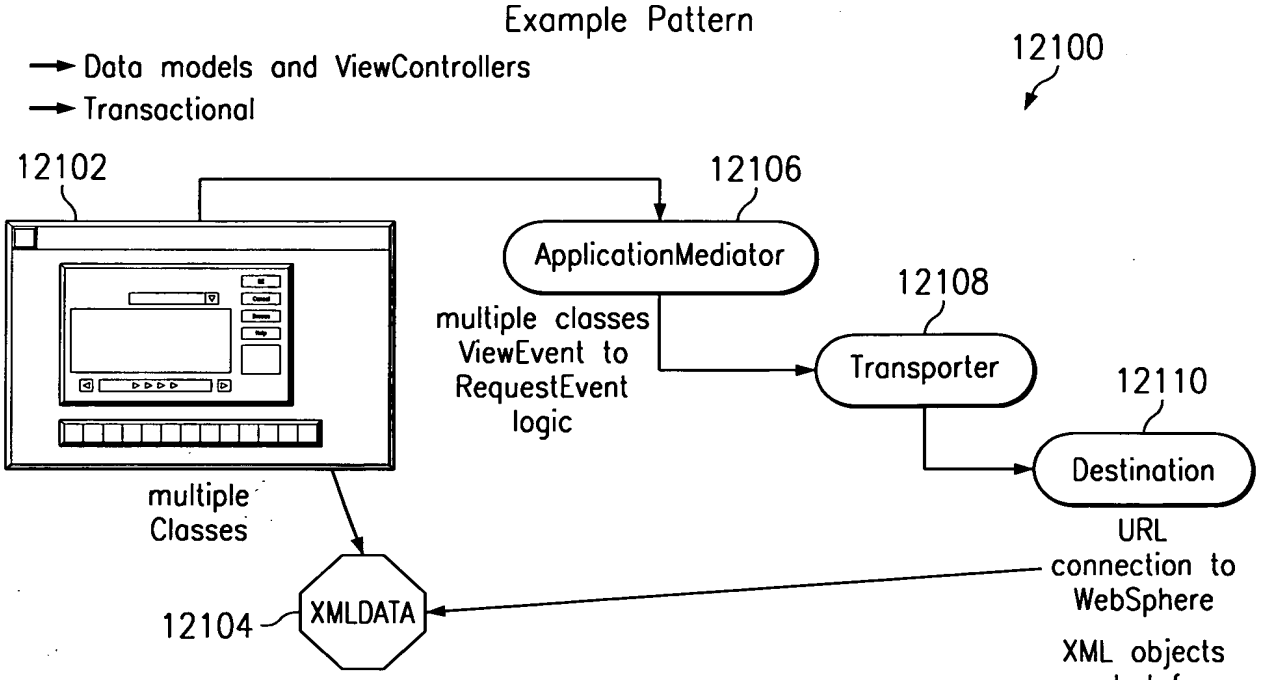



*FIG. 121*

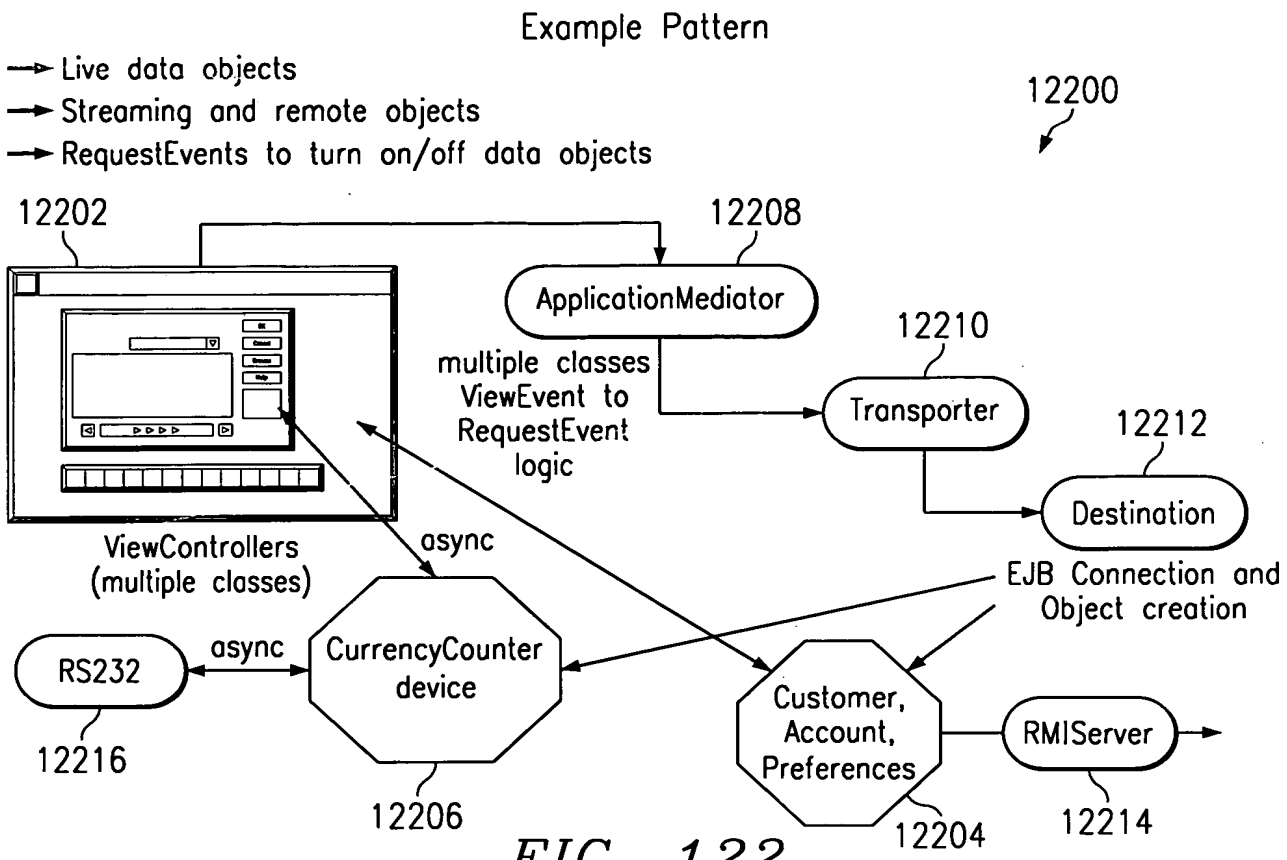



FIG. 122

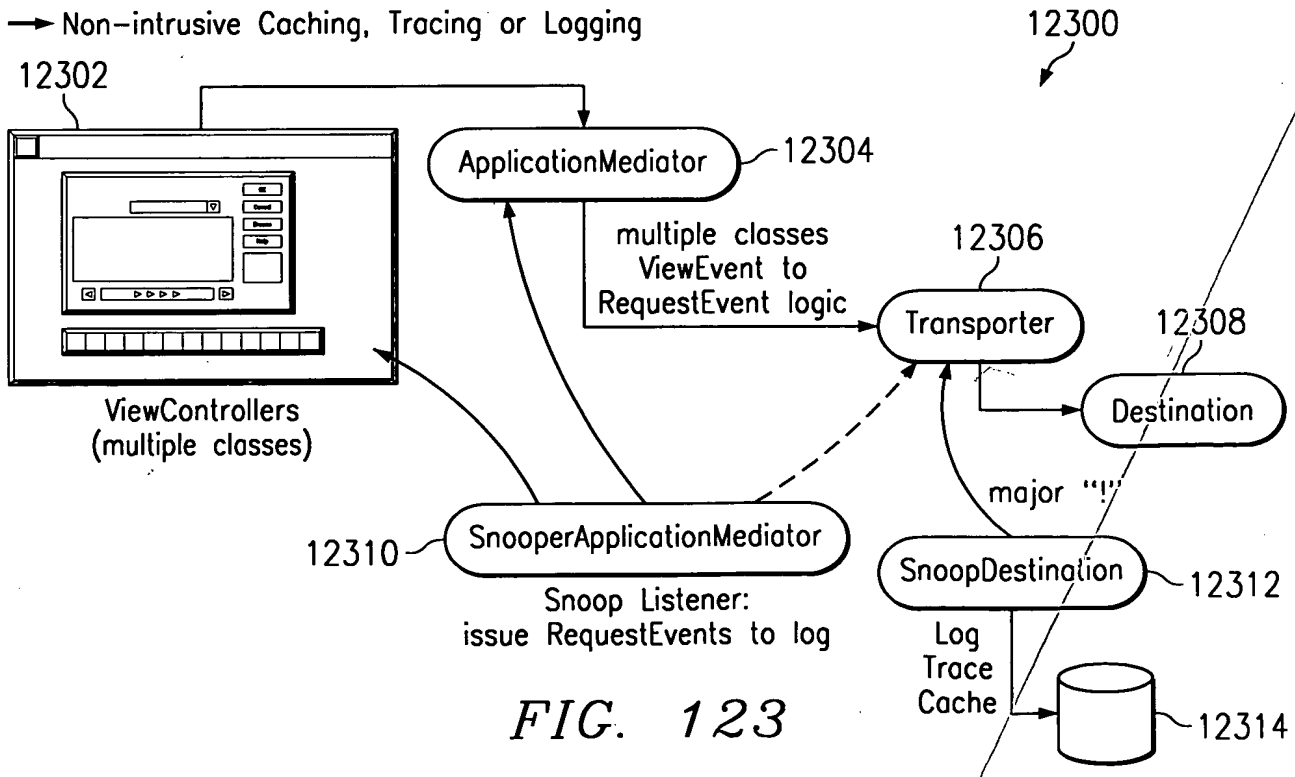



FIG. 123